Seminar zu Datenbanksystemen (SS 2005)

Green Query Optimization using Query Clustering

Elena Hensinger
Kroko1799@gmx.net
4. Juli 2005
Universität Hannover

Zusammenfassung

Da die Optimierung von Ausführungsplänen viel Rechenzeit in Anspruch nimmt, wird nach Wegen gesucht, die entstehenden Rechenkosten zu reduzieren. Eine Möglichkeit dazu bietet die Methode des Plan Recyclings durch Query Clustering, welche in diesem Seminarbeitrag vorgestellt wird. Sie arbeitet mit einem Vektorraummodell, wobei der Vektorraum durch die Anfragen charakterisierende Merkmale aufgespannt wird. Alle Anfragen mit demselben Ausführungsplan werden in Cluster mit jeweils einem Repräsentanten gruppiert, dessen vom Optimierer generierter Ausführungsplan abgespeichert wird. Neue Anfragen werden zuerst auf Ähnlichkeit mit vorhandenen Repräsentanten geprüft. Liegt eine Übereinstimmung vor, so wird, anstatt einen neuen Plan zu erzeugen, der schon optimierte Ausführungsplan des Repräsentanten dieses Clusters benutzt. Basierend auf dieser Methode wurde das Tool PLASTIC entwickelt, welches in relationale Datenbanken eingebunden werden kann.

Inhaltsverzeichnis

1 Einleitung

Die Anfrageoptimierung macht es nötig, viele verschiedene Ausführungspläne zu erstellen, miteinander zu vergleichen, und den besten Plan auszuwählen. Dabei wird in der Regel für jede Anfrage ein neuer optimierter Ausführungsplan erstellt. Insbesondere für sehr große Datenbanken oder komplexe Anfragen ist die Optimierung sehr rechenintensiv.
Eine Methode diese Rechenintensität zu verringern besteht darin, Ausführungspläne wieder zu verwenden, also Plan Recycling zu betreiben. Der erste Ansatz wäre, die textuellen Anfragen wie auch die dazugehörigen Ausführungspläne zu speichern und jede neue Anfrage mit den gespeicherten zu vergleichen. Dieses wird z. B. in den Oracle-Datenbanken durch sogenannte stored outlines gemacht. Wurden solche stored oulines erzeugt und ist die Einstellung USE_STORED_OUTLINES gesetzt, so werden die ankommenden Anfragen mit den gespeicherten Anfragen verglichen. Liegt ein photographisch gleicher SQL-Befehl vor, oder können die Eingaben aus der Anfrage statt der Bindevariablen eingesetzt werden, so wird der gespeicherte Plan benutzt. Da die Anfragen in Textform abgespeichert werden, besteht immer die Einschränkung der starken textuellen Ähnlichkeit für eine Wiederverwendung des Ausführungsplanes (siehe [GPSH02] und [H05]).
Ein anderer Ansatz ist, Anfragen nicht auf den textuellen Ähnlichkeiten zu vergleichen, sondern auf Grund ihrer strukturellen und statistischen Eigenschaften. Dieses Vorgehen basiert auf der Beobachtung, dass zwei Anfragen mit unterschiedlichen Projektionen, Selektionen, Joinprädikaten, und sogar auf unterschiedlichen Tabellen immer noch die gleiche Ausführungsplanvorlage haben und somit einen gleichen Operatorbaum benutzen können. Speichert man diese Vorlage, so muss nicht mehr jede neue Anfrage mit den gespeicherten n Anfragen, sondern nur mit den gespeicherten k Planvorlagen verglichen werden, wobei in der Regel $k \leq n$ ist (siehe [GPSH02]). Durch Benutzung eines Entscheidungsbaumes (decision tree) zur Klassifizierung neuer Anfragen werden diese Kosten noch einmal verringert, so dass sie nicht mehr proportional zur Kardinalität, sondern zur Heterogenität der Planvorlagen sind (siehe [SH04]).
Um diese Methode des Plan Recycling realisieren zu können, ist es zuallererst nötig, Anfragen in einer für Vergleiche mögliche und auf für den Ausführungsplan relevanten Daten basierenden Art zu speichern. Dieses Verfahren wird im ersten Kapitel vorgestellt. Im zweiten Kapitel werden wir uns mit dem Vergleichen von Anfragen beschäftigen. Der dritte Abschnitt ist für das Thema Clustering, Klassifizierung zu Clustern und die Ausführungsplanvorlage reserviert. Im letzten Kapitel wird das Tool PLASTIC (PLAn Selection Through Incremental Clustering) vorgestellt, welches eine Realisierung von solchem Plan Recycling ermöglicht.

2 Anfragen

2.1 Darstellung von Anfragen

Jede Anfrage kann durch typische, die Anfrage charakterisierende Merkmale (features) in Form eines Vektors (feature vector) dargestellt werden. Die Merkmale spannen damit einen Vektorraum (feature space) auf, in dem jede Anfrage platziert werden kann.
Die Auswahl der Merkmale ist ein entscheidender Schritt, welcher Auswirkungen auf die richtige Vergleichbarkeit von Anfragen, und damit auf die Zuordnung zu Ausführungsplänen hat. Ein Datenbank-Optimierer wählt den Ausführungsplan einer Anfrage mit den niedrigsten geschätzten Kosten. Verschiedene Ausführungspläne kommen aus den unterschiedlichen Arten der Joins (z. B. Nested-Loops, Merge-Join, Index-Join und Hash-Join) und deren Anordnungsreihenfolgen mit den restlichen Operationen (Selektion, Projektion und Gruppierung) zusammen. Die Kosten für die oben genannten Operationen hängen von den Eigenschaften der in der Anfrage benutzten Tabellen und der Anfrageprädikate ab. Das bedeutet, dass zur Charakterisierung einer Anfrage sowohl die für die Anfrage typischen strukturellen Informationen wie z. B. die Anzahl der Tabellen, als auch spezifische Eigenschaften der benutzten Tabellen wie z. B. das Vorhandensein und die Benutzung von Indexen mit einbezogen werden müssen. Für die zuletzt genannten dienen die Kataloge des Datenbankmanagementsystems als Informationsquelle.
Es ist wichtig, die Merkmale direkt aus der Anfrage und den vorhandenen Daten in der Datenbank entnehmen zu können. Eine Berechnung von Merkmalen kann leicht zu ihrer permanenten

Optimierung führen. Durch die Tatsache, dass die Merkmale aus statistischen, in der Datenbank vorhandenen Daten gebildet werden, passt sich die Wahl des Ausführungsplans dem aktuellen Datenstand an (siehe [SH04]).
In dem von Haritsa et al. 2002 veröffentlichten Artikel [GPSH02] wurden solche erarbeiteten charakteristischen Merkmale, die eine Anfrage spezifizieren, vorgestellt. Diese lassen sich in eine strukturelle und eine statistische Gruppe einteilen. Die Ersteren werden aus der Anfrage und den dazugehörigen schemaverwandten Metadaten entnommen, die Letzteren von den Tabellenstatistiken, die in den Systemkatalogen vorhanden sind.

2.1.1 Strukturelle Eigenschaften

Abbildung 1 dient als Veranschaulichungshilfe zum Verständnis der Merkmale. Die zwei Graphen stellen zwei Anfragen dar, wobei die Knoten die Tabellen, die Kanten die Joinprädikate repräsentieren.

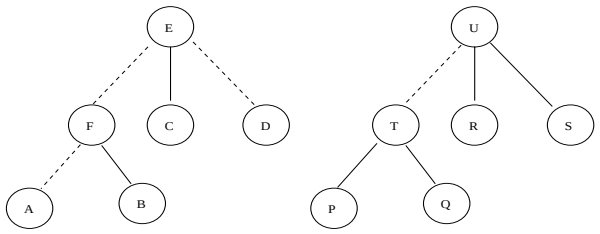

Abbildung 1: Beispiel

Degree of Table (DT): ist die Anzahl der Joinprädikate auf einer Tabelle. Zum Beispiel ist der Grad der Tabelle E = 3. Diese Information wird zum Bestimmen der Joinreihenfolge der Tabellen in dem Ausführungsplan benötigt.[1]

Degree-Sequence of a Query (DS): ist ein aus den DTs der Tabellen abgeleiteter Vektor. Der DS der beiden Beispieltabellen ist (3,2,1,1,1,1), gelesen in Präorder. Gibt es für eine Tabelle keine Joinprädikate mit einer oder mehreren anderen Tabellen, so dass ein karteisches Kreuzprodukt ausgeführt werden muss, so hat die bestimmte Tabelle den Grad 0 (siehe [SAR04]).

Number of join predicates (NJC): ist die Gesamtanzahl der Joinprädikate in der Anfrage.

Number of tables (NT): Anzahl der in der Anfrage benutzten Tabellen.

Join Predicate Index Counts (JIC) besagt, ob ein vorhandener Index für die Anfrage benutzt werden kann. Ein Joinprädikat hat eine *index characteristic* 0, 1, oder 2, wenn es 0, 1, oder 2 indexierte Attribute hat. Für jede Anfrage wird die Anzahl der Joinprädikate bezüglich der jeweiligen charakteristischen Werte gezählt.

Predicate Counts of a Table (PC): Ein Prädikat kann *SARGable*[2,3] oder *NON-SARGable* sein. Für jede Tabelle und jeden Prädikattyp werden Zähler gesetzt, um erkennen zu können, auf welchen Prädikaten ein Index denn tatsächlich benutzt werden kann, wenn er vorhanden ist. Dieses ist genau dann der Fall, wenn das damit assoziierte Prädikat SARGable ist.
In dem Beispiel sind die gestrichelten Linien SARGable, die durchgezogenen nicht. Für die Tabelle E wäre der Zähler also $(2, 1)$ mit 2 SARGable und einem nicht-SARGable Prädikat.

Index Flag of a Table (IF): zeigt für jede Tabelle an, ob ein Ausführungsplan erstellt werden kann, der nicht auf die Tabelle, sondern nur auf einen Index zugreift. Dieses ist genau dann der Fall, wenn für eine Tabelle alle Selektionsprädikate und Projektionsattribute durch den Zugriff auf einen gemeinsamen Index ausgewertet werden können.

2.1.2 Statistische Eigenschaften

Table Size (TS), die Größe der Tabelle, wird bestimmt durch das Produkt der Kardinalität der Tabelle mit der durchschnittlichen Länge der in der Tabelle vorhandenen Tupel.

[1] Ein Prädikat, der sich auf die Spalten der zwei Tabellen, die gejoint werden sollen, bezieht, wird Joinprädikat genannt.

[2] nach der Definition der System R Optimierers aus [SEL+79].

[3] *SARGable* ist die Abkürzung für Search-ARGument-able und sagt aus, ob ein Prädikat in die Form „column comparison-operator value“ gebracht werden kann (siehe [SEL+79]). Ist ein Prädikat in der WHERE-Bedingung SARGable, so kann ein bestehender Index darauf in der Ausführung benutzt werden. Nicht SARGable sind z. B. „IS NULL“, „<>“, „!<“, „NOT EXISTS“, und „NOT LIKE“, dagegen z. B. „=“ SARGable (siehe [SQL]). Die Benutzung eines Indexes lohnt sich nur bei sehr selektiven Anfragen, ansonsten ist der Zugriff über Scans schneller (siehe [DBS05]).

Effective Table Size (ETS) ist die jeweilige effektive Größe der in einem Join teilnehmenden Tabelle. Diese wird durch Abschätzungen und Statistiken aus den Systemkatalogen und den Einfluss des Herunterschiebens („push down“) der Selektionen und Projektionen im Ausführungsplan aus der Anfrage bestimmt.

2.1.3 Beispiel

Als einfaches Beispiel wird die rechts aufgeführte Anfrage genommen. Dabei sollen Indexe auf $a1$ und $b1$ in den Tabellen A und B existieren.

```
select A.a1, B.b2
from A,B
where A.a1 = B.b1
```

Die folgende Tabelle bietet eine Übersicht über die Merkmale, aus denen der Merkmalsvektor besteht, wobei im linken Teil die dazugehörigen Erklärungen zu finden sind und im rechten Teil die Werte für das obige Beispiel. Zur besseren Übersicht wurde eine Aufteilung in *globale Eigenschaften*, die sich auf die Anfrage, und die *Tabelleneigenschaften*, die sich auf die jeweiligen Tabellen beziehen, gemacht.

Globale Eigenschaften

Erklärung	Abkürzung	Wert für Beispiel-Anfrage
Anzahl der Tabellen in der Anfrage	NT	2
Vektor der Grade der Tabellen in der Anfrage	DS	(1,1)
Gesamtanzahl der Joinprädikate	NJP	1
Anzahl der Joinprädikate mit Index-Charakteristikum 0, 1, 2	$JIC[0...2]$	0, 0, 1
Anzahl der SARGable-Prädikate	$PCsarg$	1
Anzahl der nicht-SARGable-Prädikate	$PCnsarg$	0

Tabellen-Eigenschaften

Erklärung	Abkürzung	Wert für A	Wert für B
Grad der Tabelle T_i	DT_i	1	1
boolsche Angabe, ob Zugang zu T_i ausschließlich über Index möglich	IF_i	1	0
Anzahl der SARGable-Prädikate in T_i	$PCsarg_i$	1	1
Anzahl der nicht-SARGable-Prädikate in T_i	$PCnsarg_i$	0	0
Anzahl der Joinprädikate mit Index-Charakteristikum 0, 1, 2 aus T_i	$JIC_i[0...2]$	0, 0, 1	0, 0, 1
Größe der Tabelle T_i	TS_i	200.000	100.000
(geschätzte) effektive Größe der Tabelle T_i	ETS_i	200.000	100.000

Da es in der Anfrage keine Selektionen gibt, sind die effektiven Tabellengrößen gleich den geschätzten Tabellengrößen. Die Selektion durch den Join wird nicht mit in die Darstellung einbezogen, da dazu die Anordnung der Joins von vornherein bekannt sein müsste. Dieses kann jedoch nur durch einen Optimierungsprozess gegeben werden.
Es ist möglich, auf das Attribut $a1$ der Tabelle A ausschließlich über den Index zuzugreifen. Da von der Tabelle A nichts anderes mehr gefragt ist, wird der Index Flag auf 1 gesetzt. Bei Tabelle B dagegen kann nur auf $b1$ durch einen Index zugegriffen werden, obwohl auch nach $b2$ gefragt wird. Daher wird der Index Flag nicht gesetzt. Dieses wäre der Fall, wenn ein Multi-Attribut-Index $(b1, b2)$ vorhanden wäre.
Die oben genannten Merkmale, durch die eine Anfrage dargestellt werden kann, wurden in 2004 erweitert, um verschiedene Tabellenzugriffspfade (table scans, index scans) und -organisationen (clustered und B-Baum) als auch verschiedene Indextypen (unique, cluster, reverse) mit einzubeziehen. Die Zähler für PCsargable und nonSARGable- Prädikate wurden dagegen entfernt. Des Weiteren wird jetzt in einem Repräsentanten auch überprüft, ob ein vorhandener Index tatsächlich benutzt wird. Im negativen Fall wird dieser aus dem Merkmalsvektor des Repräsentanten entfernt, so dass eine bessere Zuordnung der Anfragen stattfinden kann (siehe [SH04]).

2.2 Vergleich von Anfragen durch den SIMCHECK-Algorithmus

Das Bestimmen von Ähnlichkeit oder Unähnlichkeit von zwei Anfragen erfolgt mit Hilfe der Informationen aus dem Merkmalsvektor. Durch den sogenannten SIMCHECK-Algorithmus werden die Merkmale der Anfragen miteinander verglichen und die darin benutzten Tabellen aufeinander abgebildet (siehe [GPSH02]). Je ähnlicher die Merkmale der Anfragen zueinander sind, desto wahrscheinlicher ist es, dass der Optimierer für sie gleiche Ausführungspläne generiert.
Der SIMCHECK-Algorithmus hat als Eingabe zwei Merkmalsvektoren und als Ausgabe einen booleschen Wert, der aussagt, ob die beiden Vektoren als ähnlich angesehen werden können oder nicht. Der Algorithmus teilt sich in zwei Abschnitte: den „Merkmalsvektorvergleich“ und den „Mapping Tables“-Teil.
Im ersten Teil werden die Anzahl der Tabellen in den Anfragen, die Summe der Tabellengrade, die Summe der Joinprädikate und die Eigenschaften der Joinprädikate miteinander verglichen. Sind diese Daten gleich, so wird mit dem zweiten Teil weitergearbeitet, ansonsten heißt die Ausgabe „ungleich“. Durch diese Prüfung können offensichtlich unähnliche Tabellen früh erkannt werden, um Rechenaufwand zu vermindern.
Im zweiten Teil wird nach der besten 1:1-Übereinstimmung der einzelnen Tabellen zueinander gesucht. Dadurch prüft man, ob man auf die zu vergleichenden Tabellen mit demselben Zugriffsplan zugreifen kann. Es werden auf Grund ihrer Eigenschaften zueinander passende Tabellen gesucht und mit Hilfe einer Distanzfunktion auf den Tabellengrößen und den effektiven Tabellengrößen miteinander verglichen. Gibt es mehrere Möglichkeiten der Abbildung, so wird die mit der kleinsten Distanz gewählt. Für die Gesamtdistanz werden die Mindestdistanzen der Tabellen summiert. Unterschreitet der berechnete Wert einen Schwellwert, so sind die Anfragen ähnlich, ansonsten unähnlich. [4]
Im Folgenden ist der SIMCHECK-Algorithmus als Pseudocode wiedergegeben und näher erläutert, wobei die Abkürzungen in den Tabellen der globalen- bzw. Tabellen-Eigenschaften definiert worden sind und hier in den Kommentaren noch einmal benannt werden:

SIMCHECK-Algorithmus

Input: Anfrage Q1, Anfrage Q2
Output: „similar“ oder „not similar“

// Prüfe, ob Anzahl der Tabellen gleich ist
1. IF NT(Q1) != NT(Q2) RETURN (not similar); /// NT = Anzahl der Tabellen (number of tables)

// Prüfe die Level-Semantik
2. IF NOT {DS(Q1) = DS(Q2) AND /// DS = Vektor der Grade der Tabellen (degree-sequence of a query)
NJP(Q1) = NJP(Q2) AND /// NJP = Anzal der Joinprädikate (number of join-predicates)
PCsarg(Q1)+PCnsarg(Q1) = PCsarg(Q2)+PCnsarg(Q2) /// Die Zugriffsmöglichkeiten auf die Prädikate müssen gleich sein
} then GO TO Step 3 else GO TO Step 4;

3. RETURN(not similar);

// Finde die beste Zuordnung zwischen den Tabellen
4. FOR jede Gruppe g von Tabellen desselben Grades
$R1 = T_1^1, T_1^2, ..., T_1^k \quad R_1 \subseteq Q_1$ /// zu T_1^1 gehören alle Tabellen der ersten Anfrage, deren Anzahl der Join-Elemente = 1 ist, usw.
$R2 = T_2^1, T_2^2, ..., T_2^k \quad R_2 \subseteq Q_2$

// Finde die beste Zuordnung zwischen $R1$ und $R2$ mit der minimalen aggregierten Distanz $mindist_g$
// bezogen auf die paarweise Distanzfunktion

$$dist_{i,j}(T_1^i, T_2^j) = \frac{w_1 \cdot |TS_1^i - TS_2^j| + w_2 \cdot |ETS_1^i - ETS_2^j|}{max(TS_1^i, TS_2^j)}$$

[4] Obwohl nicht explizit erwähnt, werden der Join-Index-Count (JIC) und der Index Flag (IF) für die Abbildung der Tabellen benutzt.

// hierbei ist TS die Größe und ETS die effektive Größe der Tabelle.
// Berechne die Distanz zwischen den Anfragen
5. $TotalDist = \sum_{g \in G} mindist_g$

6. IF $TotalDist > Threshold$ RETURN (not similar);

7. RETURN(similar);

Schritte 1 - 3:

Die Bestimmung, ob zwei Tabellen zueinander passend sind, basiert unter anderem auf der Gleichheit der Summe der Tabellengrade in der Anfrage, der Summe der Joinprädikate und ihrer Eignung zur Benutzung eines Indexes. Es ist wichtig zu vergleichen, wie viele Prädikate der Tabellen SARGable sind und wie viele nicht. Davon hängt der vom Optimierer berechnete Ausführungsplan ab. Auf zwei Tabellen, deren Prädikatzähler $(2,1)$[5] und $(1,2)$ sind, wird nicht in derselben Weise zugegriffen werden können. Ein vorhandener Index kann auf nonSARGable-Prädikaten nicht benutzt werden. Daher können solche Anfragen nicht gleich sein.
Einen noch stärkeren Einfluss auf den Ausführungsplan haben die Indexzähler. Existieren Indexe auf Joinprädikate in der einen Anfrage, aber nicht in der anderen, so werden die Pläne in der Regel unterschiedlich sein. Bei Indexen auf beide Joinprädikate und starker Selektivität der Tabelle kann ein Index-Join angewendet werden. Bei einer Indexierung von nur einem Joinprädikat kann ein Indexzugriff statt eines Tablescans erfolgen. Diese Optionen sind ohne Indexierung nicht vorhanden.

Schritte 4 - 7:

Es gibt Fälle, in denen die vom Optimierer berechneten Ausführungspläne trotz der Gleichheit der Anzahl der Joins und der Anzahl und Art der Prädikate für zwei Anfragen unterschiedlich sind. Dieses hat statistische Faktoren wie z. B. Tabellengrößen als Grund. Diese Faktoren werden von der Distanzfunktion abgedeckt. Es wird nach einer 1:1-Zuordnung für die als zueinander passend bestimmten Tabellen gesucht. Die Größen und die effektiven Größen der Tabellen T_1^i und T_2^j werden miteinander durch die Distanzfunktion

$$dist_{i,j}(T_1^i, T_2^j) = \frac{w_1 \cdot |TS_1^i - TS_2^j| + w_2 \cdot |ETS_1^i - ETS_2^j|}{max(TS_1^i, TS_2^j)}$$

verglichen. Dabei steht T_1^i für die *i-te* Tabelle der ersten Anfrage und T_2^j für die *j-te* Tabelle der zweiten Anfrage, die aufeinander abgebildet werden sollen. Je größer die Distanz dabei wird, desto kleiner ist die Ähnlichkeit. Dabei sind w_1 und w_2 Gewichtungsfaktoren, deren Summe gleich 1 ist. Typischerweise wird w_1 größer gesetzt als w_2, da der Einfluss der tatsächlichen Größen der Tabellen meisten viel stärker ist als der der effektiven Größen. Empirische Messungen ergaben gute Ergebnisse für $w_1 = 0,7$ und $w_2 = 0,3$, wobei die Leistung mit $\pm 0,1$ nur marginal beeinträchtigt wird.
Wie man sieht, werden nicht nur die Kardinalität der Tabellen, sondern auch die Anzahl der Tupel durch die effektive Größe in die Rechnung mit einbezogen. Der Grund dafür ist der, dass für einen Ausführungsplan die Gesamtsumme der Daten, die abgerufen und in den Zwischenspeichern gespeichert werden, ausschlaggebend ist. Damit haben die Längen der Tupel der Projektionen und Selektionen bei gleich bleibenden übrigen Faktoren einen Einfluss auf die Wahl des Planes.
Durch die Normalisierung gilt $0 \leq dist_{i,j} \leq 1$, so dass die Summe *dist* von der Anzahl der Tabellen in der Anfrage nach oben begrenzt wird. Gibt es mehrere Möglichkeiten, Tabellen einer Anfrage aufeinander abzubilden, so wird die Abbildung mit der kleinsten Distanz gewählt. Eine Ähnlichkeit zweier Anfragen ist genau dann gegeben, wenn *dist* einen bestimmten benutzerdefinierten Schwellwert unterschreitet.
Dieser Schwellwert bestimmt im weiteren Verlauf die Größe der Cluster, die aus den Anfragen gebildet werden. Sein Wert ist von der Größe der Datenbank abhängig: Für Datenbanken mit großen Tabellen ist ein kleiner Schwellwert passend, da der Plan des Optimierers stark von den Änderungen

[5] mit zwei SARGable und einem nicht-SARGable Prädikaten

der Selektivitäten anhängt. Werden die Cluster groß in Bereichen mit starken Selektivitäten, so wächst die Wahrscheinlichkeit der Zuordnung einer Anfrage zu einem Cluster mit einem falschen Ausführungsplan. Für den TPC-H-Benchmark und die dazugehörige Datenbank wurde für Cluster gleicher Größe der Wert 0,01 als gut ermittelt (mehr dazu im Kapitel „Clustering“).
Durch die Gruppierung der Tabellen nach ihrem Grad muss für eine mögliche Abbildung nicht mehr jede Tabelle mit jeder anderen verglichen werden, so dass die Komplexität nicht mehr $O(n!)$ ist. Wenn die Tabellen der beiden Anfragen $Q1$ und $Q2$ aufgeteilt sind in Gruppen $p_1, p_2, ..., p_k$ nach ihren Graden, wird die Komplexität der Zuordnung $O(p_1! + p_2! + ... + p_k!)$, da nur noch die zueinander passenden Gruppen betrachtet werden müssen. Im Durchschnitt ist dies weit weniger als die $O(n!)$ einer *n:n*-Zuordnung. Zusätzlich dazu wird die Rechenzeit dadurch reduziert, dass nicht alle möglichen Tabellen, sondern nur die im ersten Teil des Algorithmus als die zueinander passenden, verglichen werden.
Es gibt Fälle, in denen sich die Merkmalsvektoren von zwei Anfragen unterscheiden, der Datenbank-Optimierer trotzdem den selben Ausführungsplan liefert, da die Schätzung der zu bearbeitenden Daten ähnlich groß ist. Dieses ist z. B. der Fall bei den Anfragen 1) und 2):

```
1) select * from part
```

```
2) select p_brand, p_name, p_mfgr
from part
where p_size = 4 and p_brand='Brand#15'
```

Ohne Kenntnisse über die Tabellen zu haben kann man davon ausgehen, dass sich die Merkmalsvektoren dieser beiden Anfragen in den effektiven Tabellengrößen unterscheiden. Desweiteren in der Anzahl der SARgable und nicht-SARGable Prädikate, da in der zweiten Anfrage nicht alle, dafür einige bestimmte Prädikate abgefragt werden.
Diese Situation ist mit der Entscheidung, über den Merkmalsvektor und nicht z. B. über den vom Optimierer bestimmten Plan zu vergleichen, verbunden. Der Vorteil bei dem hier vorgestellten Verfahren ist, dass die spätere Gruppierung aller ähnlichen Anfragen durch ihre Merkmalsähnlichkeit klar definiert ist, während in der Gruppierung über den Plan die Gruppen inhomogen wären. Das würde dazu führen, dass bei einem Vergleich von Anfragen immer noch mit jeder gespeicherten Anfrage verglichen werden müsste, anstatt mit einem Repräsentanten, welches keine Verbesserung der Laufzeit mit sich brächte.

3 Clustering

Der nächste Schritt zur Wiederverwendung von Plänen besteht darin, Anfragen, die die gleichen Ausführungspläne haben, zu gruppieren. Dieses geschieht durch das Clustering, welches allgemein Daten in Gruppen segmentiert, wobei die Gruppen vorgegeben sein können oder erst während des Ablaufs gebildet werden, ihre Anzahl vorgegeben wird oder nicht. Gutes Clustering zeichnet sich dadurch aus, dass eine starke Ähnlichkeit der Objekte innerhalb des Clusters und eine starke Unähnlichkeit dieser Objekte zu den Objekten ausserhalb des Clusters besteht.
In der Abbildung 2 sieht man auf der linken Seite mehrere ungeordnete Objekte, die im mittleren Bild nach ihrer geographischen Position gruppiert werden, in rechten Bild nach einer Eigenschaft, (siehe [WDM04]). Wird die Zahl der zu erzeugenden Cluster nicht angegeben, so werden hierarchische Verfahren benutzt, im anderen Fall die partitionierenden.

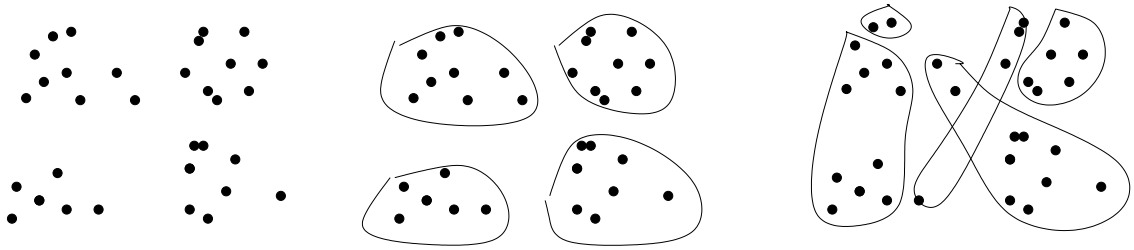

Abbildung 2: Beispiele für Clustering

3.1 Hierarchisches Clustering

Die hierarchischen Verfahren zeichnen sich dadurch aus, dass auf dem untersten Level jedes Objekt einen eigenen Cluster formt, im obersten Level nur noch ein Cluster existiert. Diese Verfahren können wiederum in zwei Gruppen unterschieden werden: agglomerative (bottom-up) und divisive (top-down) Verfahren. Die Hierarchien, welche durch das hierarchische Clustering entstehen, lassen sich durch eine Baumdarstellung, das sogenannte Dendrogramm, darstellen (vgl. Abbildung 3). Dabei sind die Blätter in dem Dendrogramm die Cluster mit jeweils einem Objekt, die Wurzel der Cluster mit allen Objekten. Interne Knoten sind die Cluster, die durch das Zusammensetzen der einzelnen Ebenen entstehen. Auf der X-Achse werden die untersuchten Objekte und auf der Y-Achse die Distanz zwischen diesen (Ähnlichkeit) aufgetragen. Die gröbste Partitionierung findet man bei der maximalen, die feinste bei der minimalen Distanz.

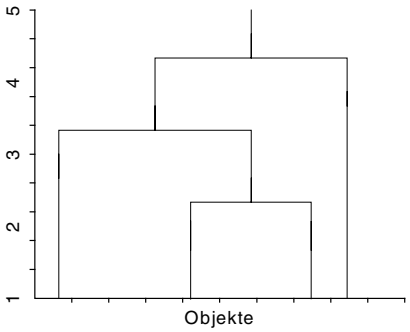

Abbildung 3: Beispiel für ein Dendrogramm

3.1.1 Hierarchisches Agglomeratives Clustering

Beim hierarchischen agglomerativen Clustering (HAC) findet der folgende Ablauf statt:

1. Erzeuge für jedes Objekt auf dem untersten Level je einen Cluster.
2. Fasse die beiden Cluster mit der kleinsten Distanz zu einem Cluster zusammen.
3. Berechne den Abstand des neuen Clusters zu den vorhandenen.
4. Wenn insgesamt mehr als ein Cluster vorhanden ist, mache weiter ab Schritt 2.

Der Unterschied der verschiedenen Arten des agglomerativen Verfahrens liegt in der Berechnung der Distanz zwischen zwei Clustern (siehe [EID00] und [STAT]).
Die Objekte können als mehrdimensionale Vektoren dargestellt werden, deren Distanz zueinander, also die Ähnlichkeit bzw. Unähnlichkeit, u. a. mit

1. der Euklidischen Distanz: distance(x,y) $= \{\sum_i |x_i - y_i|^2\}^{\frac{1}{2}}$
2. der City-block (Manhattan) Distanz: distance(x,y) $= \sum_i |x_i - y_i|$
3. Tschebyscheff-Distanz: distance(x,y) $= \text{Max}|x_i - y_i|$

berechnet werden kann.
Des Weiteren gibt es mehrere Möglichkeiten, die Distanz zwischen Clustern mit mehreren Objekten zu berechnen. Dazu hat man u. a. die folgenden Alternativen zur Auswahl:

1. größte Distanz (complete link): es zählt der größte Abstand zwischen zwei Objekten in den Clustern.
2. kleinste Distanz (single link): es zählt der kleinste Abstand zwischen zwei Objekten in den Clustern.
3. Vergleich zwischen den Zentroiden der Cluster, wobei ein Zentroid der berechnete Mittelpunkt des Clusters ist (Schwerpunkt). Dieser muss keinem tatsächlichen Objekt im Cluster entsprechen.

Im Folgenden sind zwei Verfahren im Vergleich dargestellt: Benutzt man das single-link-Verfahren, so entstehen längliche, oft verkettete Cluster, siehe Abbildung 4. Ganz anders geformte Cluster

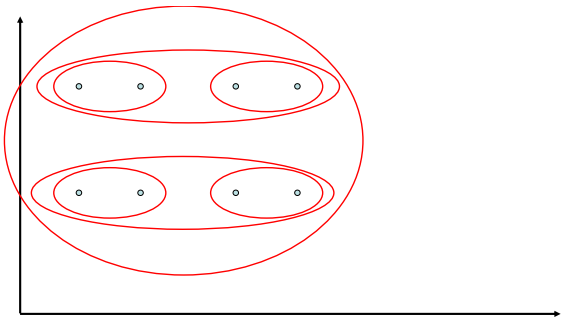

Abbildung 4: HAC mit single link

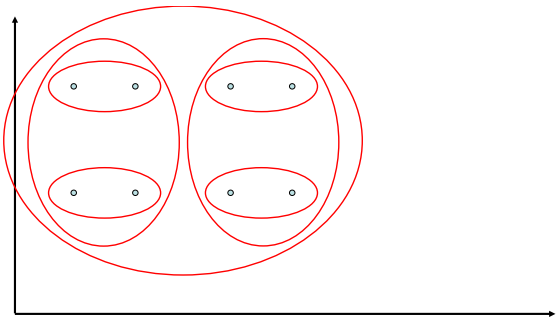

Abbildung 5: HAC mit complete link

entstehen dagegen bei Benutzung der größten Distanz (complete link), wie in Abbildung 5, beides aus [MR05], zu sehen.
Die Komplexität des agglomerativen hierarchischen Clusterings ist im ersten Schritt, in dem jedes Objekt mit jedem anderen verglichen werden muss, $O(n^2)$. In den nächsten Iterationen wird die Distanz des neuen Clusters zu den Alten berechnet. Wird dies in einer konstanten Zeit gemacht, bleibt die Komplexität $O(n^2)$, ansonsten wird sie $O(n^2 \log n)$ oder sogar $O(n^3)$ (siehe [MR05]). Dieses ist auch der große Nachteil des Verfahrens. Die Vorteile besteht darin, dass man das Ergebnis graphisch gut in Dendrogrammen darstellen kann und die Anzahl der benötigten Cluster nicht vorgegeben werden muss, sondern zur Laufzeit bestimmt wird.

3.1.2 Hierarchisches Divisives Clustering

Das divisive Verfahren hat den folgenden Ablauf:

1. Starte mit dem obersten Cluster, welcher alle Objekte enthält.
2. Suche das Objekt mit der größten Entfernung zu allen anderen Objekten im Cluster. Bilde damit einen neuen Cluster als Knoten des Baumes.
3. Berechne für jedes Objekt im Ausgangscluster: (durchschnittlicher Abstand zu allen Objekten im neuen Cluster) - (durchschnittlicher Abstand zu allen Objekten im Ausgangscluster).
4. Ordne das Objekt mit dem größten Ergebnis in den neuen Cluster.
5. Fahre so lange fort, bis die Differenz negativ ist. Dann ist der neue Cluster vollständig.
6. Führe das Splitting rekursiv auf beide entstandenen Cluster durch, bis jedes Objekt sich auf unterster Ebene in einem eigenen Cluster befindet.

Das divisive Clustering ist mit mehr Rechenaufwand verbunden und nicht so populär wie das agglomerative hierarchische Clustering.

3.2 Partitionierendes Clustering

Bei den partitionierenden Verfahren besteht die Aufgabe, n Objekte zu k Clustern zuzuordnen. Als Beispiele werden die k-means-Methode und der Leader-Algorithmus näher erklärt.

3.2.1 k-means

Der k-means-Algorithmus benutzt die Zentroide der Cluster, um die zu ordnenden Objekte damit zu vergleichen. Der Ablauf des Algorithmus ist wie folgt (siehe [WDM04]):

1. Zur Initialisierung werden den k Clustern beliebige Durchschnitte (Zentroide) gegeben und die Objekte daraufhin in die Cluster aufgeteilt.
2. Daraufhin werden die neuen Durchschnitte berechnet.
3. Jetzt werden die einzelnen Objekte in die Cluster verschoben, bei denen der Abstand vom Objekt zum Durchschnitt des Clusters am kleinsten ist.

4. Wiederholung ab Schritt 2.

Eine Abbruchbedingung stoppt den Ablauf, wenn z. B. eine bestimmte Anzahl von Iterationen durchgeführt wurden oder die Veränderung der Zentroide nur noch gering ist.
Ein Beispiel zu dem k-means-Algorithmus:

Gegeben sind die Zahlen {1, 21, 15, 79, 9}, die in zwei Cluster C_1 und C_2 einsortiert werden sollen. Im ersten Schritt werden als Durchschnitte $C_{1,d} = 1$ und $C_{2,d} = 21$ angegeben. Die euklidische Distanz wird berechnet, um die Objekte den Clustern zuzuordnen.

$$\begin{array}{ll} D_{1,C_1} = 0 & D_{1,C_2} = 20 \\ D_{21,C_1} = 20 & D_{21,C_2} = 0 \\ D_{15,C_1} = 14 & D_{15,C_2} = 6 \\ D_{79,C_1} = 78 & D_{79,C_2} = 58 \\ D_{9,C_1} = 8 & D_{9,C_2} = 12 \end{array}$$

Die Ausgangscluster sind damit $C_1 = \{1, 9\}$ und $C_2 = \{21, 15, 79\}$, die dazugehörigen neuen Clusterdurchschnitte sind:

$$C_{1,d} = \frac{1+9}{2} = 5 \qquad C_{2,d} = \frac{21+15+79}{3} = \frac{115}{3} = 38,\overline{3}$$

Jetzt wird die Distanz der einzelnen Objekte zu den Clusterdurchschnitten neu berechnet und die Zuordnung aktualisiert. Die neuen Cluster sind dann $C_1 = \{1, 15, 9\}$ und $C_2 = \{21, 79\}$. Der Gesamtablauf der Berechnung ist in der folgenden Tabelle dargestellt:

It.-schritt	Cluster	Clusterdurchschnitte
0.	$C_1 = \{\}$, $C_2 = \{\}$	$C_{1,d} = 1$, $C_{1,d} = 21$
1.	$C_1 = \{1, 9\}$, $C_2 = \{21, 15, 79\}$	$C_{1,d} = 5$, $C_{2,d} = 38,\overline{3}$
2.	$C_1 = \{1, 15, 9\}$, $C_2 = \{21, 79\}$	$C_{1,d} = 8,\overline{3}$, $C_{2,d} = 50$
3.	$C_1 = \{1, 21, 15, 9\}$, $C_2 = \{79\}$	$C_{1,d} = 11,5$, $C_{2,d} = 79$
4.	$C_1 = \{1, 21, 15, 9\}$, $C_2 = \{79\}$	$C_{1,d} = 11,5$, $C_{2,d} = 79$

Nach der dritten Iteration ergibt sich keine Änderung in den Clustern, so dass damit die endgültige Zuordnung erreicht ist.
Die Komplexität dieses Algorithmus ist nach [MR05] $O(i \cdot k \cdot n \cdot m)$, wobei i die Anzahl der Iterationen, k die Anzahl der Cluster, n die Anzahl der Objekte und m die Dimension der Objekte sind. Dabei sind k und m konstant, der Wert für i hängt u. a. von den Werten k, n und den konkreten Daten ab. Der k-means-Algorithmus konvergiert typischerweise nach wenigen Iterationen, so dass die Laufzeit unter den Umständen, dass die Anzahl der zu clusternden Objekte n viel größer als die Anzahl der Cluster k ist, annähernd $O(n)$ ist. Für eine große Anzahl von Clustern verschlechtert sich die Laufzeit allerdings. Im schlechtesten Fall ist $k = n$, so dass jedes Objekt mit allen anderen Objekten verglichen werden muss. Dann ist die Laufzeit annähernd $O(n^2)$.
Der Vorteil von k-means ist die unter bestimmten Gegebenheiten günstige Laufzeit, das intuitive Vorgehen und die Bildung kompakter und runder Cluster. Der Nachteil ist die Tatsache, dass ein gutes Ergebnis sehr von der Initialauswahl der Zentroide der Cluster abhängt. Eine verbreitete Methode ist die zufällige Wahl der Zentroide. Ein weiteres Problem ist die Bestimmung der richtigen Anzahl k der Cluster, die zu Beginn des Algorithmus angegeben werden muss. Dazu werden oft mehrere Durchläufe mit unterschiedlichen Größen für k gemacht, um das beste Ergebnis zu wählen.

3.2.2 Leader-Algorithmus

Während es bei dem k-means-Verfahren nötig ist, alle zu clusternden Objekte zu Beginn der Zuordnung zu kennen, werden die Objekte bei dem Leader-Algorithmus nacheinander bearbeitet, so dass keine Kenntniss über alle Objekte gegeben sein muss. Nach [EID00] ist der Ablauf wie folgt:

1. Das erste Objekt wird zum Klassenführer, dem Leader, eines Clusters.

2. Neue Objekte werden mit den vorhandenen Leadern verglichen, indem ihre Distanzen berechnet werden. Ein Objekt wird dem Cluster mit der kleinsten Distanz zum Leader zugeordnet.

3. Kann ein Objekt keinem Cluster zugeordnet werden, da die Distanz einen Parameter ρ überschreitet, so wird ein neues Cluster mit diesem Objekt als Leader gebildet.

4. Der Algorithmus endet, wenn alle Objekte abgearbeitet sind, wobei nicht zugeordnete Objekte unbehandelt bleiben.

Die gebildeten Cluster hängen von der Reihenfolge der eintreffenden Objekte ab, so dass unter Umständen mehr als die minimal nötige Anzahl der Cluster gebildet wird.
Für die Gruppierung von Anfragen mit dem Ziel des Plan Recyclings bietet sich der Leader-Algorithmus aus Gründen der Laufzeit von $O(k \cdot n)$ mit k Clustern und n Anfragen an, die sich insbesondere in Umgebungen mit vielen Anfragen bemerkbar macht. In der Regel ist $k \ll n$, da jeder Leader u. U. Hunderte von Anfragen vertreten kann. Damit ist die Laufzeit linear in Abhängigkeit von der Anzahl der Cluster (siehe [GPSH02]).
Das Erstellen eines neuen Clusters beim Leader-Algorithmus ist nicht mit einer Überarbeitung aller bisher vorhandenen Cluster verbunden. Des Weiteren ist der Algorithmus sehr einfach zu handhaben, da z. B. keine komplexen Datenstrukturen mit bearbeitet werden müssen.
Eine Eigenschaft der Cluster, die mit diesem Algorithmus und der Distanzfunktion $dist_{i,j}$ entstehen, ist ihre Form als Ellispoiden im Ausführungsplanraum der Anfragen. Der Grad der Ausdehnung des Clusters in eine Dimension hängt von der Tabelle in dieser Dimension ab. Der Nenner der Distanz-Funktion ist für jede Abbildung der Tabellen unterschiedlich. Dadurch entstehen im mehrdimensionalen Raum Ellipsoiden.
Für die Benutzung des Clustering im Plan Recycling wird als Zentroid eines Clusters sein Repräsentant benutzt. Zu diesem werden die Distanzen der neu ankommenden Anfragen berechnet.

3.3 Größe und Anzahl der Cluster

Die Größe und die Anzahl der Cluster, zu denen Anfragen gruppiert werden, wird durch den Schwellwert der Distanzfunktion bestimmt. Diese besagt, wie groß der Radius des Clusters werden soll. Der Radius ist die maximale Entfernung eines jeden Elementes zum Zentrum des Clusters. Dadurch wird bestimmt, wie groß die Ähnlichkeit von zwei Anfragen sein muss, um zu einem Cluster zu gehören.
Je kleiner der Schwellwert ist, desto genauer ist die Zuordnung zu den vorhandenen Clustern und desto kleiner ist die Wahrscheinlichkeit einer Falschzuordnung. Allerdings wächst damit die Rechenzeit für die Klassifizierung, da die Anzahl der Cluster steigt.

Anfragevorlage Q_2:

```
select
 s_acctbal, s_name, n_name, p_partkey,
 p_mfgr, s_address, s_phone, s_comment
from
 part p, supplier s, partsupp ps,
 nation n, region r
where
 p_partkey = ps_partkey and
 s_suppkey = ps_suppkey and
 p_size = :1 and p_type like :2 and
 s_nationkey = n_nationkey and
 n_regionkey = r_regionkey and
 r_name = :3 and ps_supplycost = :4
```

Arbeitet man mit fest eingestellten Größen der Cluster, so wird, nachdem ein bestimmter Cluster voll ist, ein weiterer aufgemacht, der allerdings denselben Ausführungsplan P_i hat. Dieser Overhead wird im schlechtesten Fall bei k Clustern zu $k-1$ redundanten Merkmalsvektoren und Planvorlagen führen. Um diese Overheads zu vermeiden, werden alle Vorlagen verglichen und alle gleichen Vorlagen einem einzigen Plan P_i zugewiesen (siehe GPSH02).
Dieses Vorgehen kann auch zur Lösung des Problems führen, dass zu einem Ausführungsplan mehrere verschiedene Cluster gehören können. Aus der Entscheidung, nach Merkmalen der Anfragen zu clustern kann es passieren, dass sehr unterschiedliche Anfragen den gleichen Ausführungsplan generiert bekommen. Damit ergibt sich eine Beziehung N:1 von N Clustern im Merkmalsraum zu einem Plan im Planraum.
Der Vorteil solch einer Clusterung ist das Übereinstimmen der Objekte sowohl im Merkmals- als auch im Planraum, so dass eine eindeutige Zuordnung von Anfragen zu Clustern möglich ist.

In Versuchen mit Anfragen zeigte sich, dass unterschiedlichen Ausführungspläne für Anfragen nicht gleichmässig im Ausführungsplanraum verteilt sind. Dieses kann an Hand von Anfragevorlagen visualisiert werden. Bei einer Anfragevorlage werden manche oder alle Konstanten im *where*-Teil der Anfrage durch Variablen, sogenannte bind variables, ersetzt. In der aus dem TPC-H Benchmark vereinfachten Anfragevorlage Q_2 werden die Eigenschaften Kontostand (acctbal), Name (s_name), Land (n_name), Nummer (p_partkey) und Hersteller (p_mfgr) eines Produktes, die Adresse (address), Telefonnummer (phone) und Bemerkung (comment) aller Lieferanten aus einer bestimmten Region (r_name), die Artikel von einer bestimmten Größe (size) und Typ (type) zu einem bestimmten Preis (supplycost) liefern können, ausgegeben.
Durch die Variierung der Variablen :1, :2 und :4 und damit der Selektivitäten der Tabellen PART und PARTSUPP bei gleichbleibenden übrigen Faktoren entsteht das zweidimensionale *Plandiagramm* aus Abbildung 6.

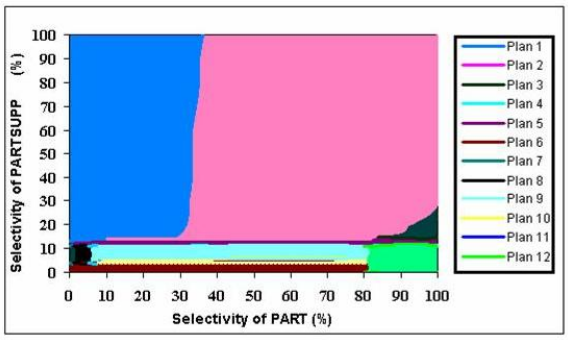

Abbildung 6: Plandiagramm

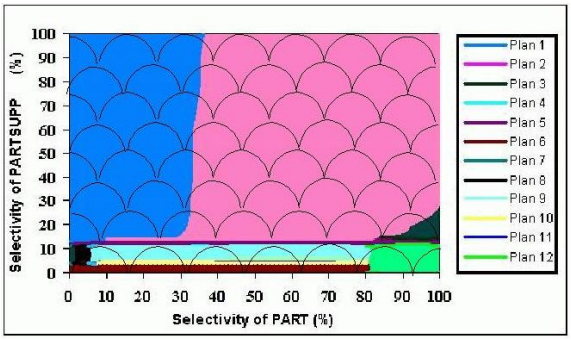

Abbildung 7: Cluster im Plandiagramm

In dem Plandiagramm sieht man, dass für verschiedene paarweise Zuordnungen der Selektivitäten der beiden Tabellen unterschiedliche Ausführungspläne vom Optimierer generiert werden. Insgesamt gibt es zwölf unterschiedliche Pläne, wobei die Pläne 1 und 2 den meisten Raum im Planraum abdecken. Das Plandiagramm hat immer genau so viele Dimensionen, wie es Tabellen mit Selektionsprädikaten in der Anfrage gibt.
Die Abbildung 7 zeigt den Planraum der Anfrage Q_2 beispielhaft abgedeckt durch gleich große semi-elliptische Cluster, die durch die benutzte Distanzfunktion entstehen (siehe [H03]).
Der gesamte Planraum teilt sich in zwei Regionen auf: die *high-volatility*-Regionen mit sich oft verändernden Ausführungsplänen (im unteren Teil der Abbildung 6), und die *low-volatility*-Regionen mit sich nur selten verändernden Plänen (Bereiche mit Plänen 1 und 2). Die häufige Änderung der Ausführungspläne ist typischerweise in stark selektiven Regionen anzutreffen.
Sind nun alle Cluster gleich groß, so haben die high-volatility-Regionen unterbesetzte Cluster, die low-volatility dagegen überflüssige Cluster mit gleichen Ausführungsplanvorlagen. Daher ist es sinnvoll, mit unterschiedlich großen Clustern in den verschiedenen Regionen des Planraumes zu arbeiten, nämlich vielen kleinen Clustern in den high- und wenigen großen Clustern in den low-volatility-Regionen wie in Abbildung 8 zu sehen.
Die die x- und y-Achsen repräsentieren die Selektivitäten der beiden betrachteten Relationen PART und PARTSUPP, die Punkte stellen die Clusterrepräsentanten dar. In den Regionen mit starken Selektivitäten auf den einzelnen Tabellen, also entlang der Achsen, sind die Cluster sehr klein. Ihre Größe und die Anzahl entspricht den sich oft verändernden Ausführungsplänen.
Je mehr Tupel ausgewählt werden, also die Selektivität schwächer wird, desto größer und seltener werden die Cluster. Dieses entspricht der Tatsache, dass die Ausführungspläne der Anfragen mit diesen Konstellationen der Selektivitäten sich nur selten verändern.
Durch die Anpassung der Schwellwerte der Distanzfunktion und damit der Radien der Cluster an die unterschiedlichen Planregionen konnten in Versuchen die Fehler in der Voraussage des Ausführungsplanes um ca. 50 % reduziert werden.
Kombiniert man ein Plandiagramm mit den jeweiligen Kosten der Pläne, so entsteht ein Plan-Kosten-Diagramm. In der Abbildung 9 ist ein Beispiel für einen Planraum zu sehen, in dem es insgesamt 11 Planregionen gibt. Die Größe der Punkte stellt die relativen Kosten der Pläne dar (siehe [SAR04]).

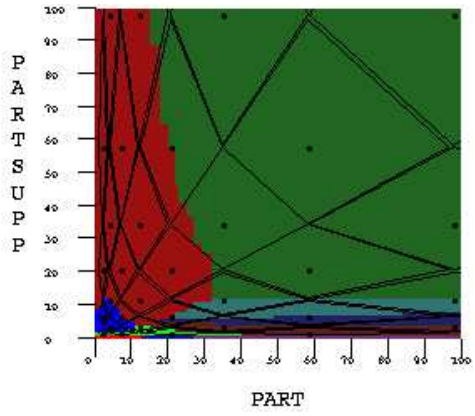

Abbildung 8: Variable-Sized Clustering

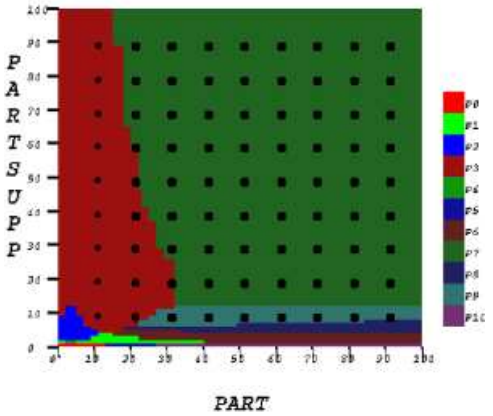

Abbildung 9: Plan-Kosten-Diagramm

Bei der Zuordnung von Anfragen zu Clustern mit einem bestimmten Ausführungsplan entstehen Falschzuordnungen immer dann, wenn ein Bereich im Clusterraum von mehr als einer Planvorlage im Planraum abgedeckt wird. Da der Ausführungsplan der Anfrage immer mit dem des Clusterrepräsentanten assoziiert wird, weicht in diesem Fall der vom Optimierer gewählte Plan für diese Anfrage ab. Der Fehler steigt mit der Differenz der Effektiven Größen der Anfrage und des Repräsentanten, da dann die Anfrage am Rand des Clusters plaziert wird. Die Fehlerwahrscheinlichkeit im Planraum hängt mit der Größe der Cluster zusammen, und damit von der Wahl des Schwellwertes der Differenzfunktion ab.

3.4 Klassifizierung von Anfragen zu Clustern

Für jedes Cluster wird für die Anwendung des Plan Recyclings der Repräsentant und der dazugehörige, vom Optimierer bestimmte, Ausführungsplan gespeichert. Das Vergleichen bzw. Einordnen von Anfragen zu Clustern sollte möglichst schnell und exakt erfolgen. Das kann direkt durch den Leader-Algorithmus geschehen, welches jedoch bei einer großen Anzahl von Clustern sehr rechenaufwändig wird, da die Anfrage mit jedem Repräsentanten verglichen werden muss. Daher bietet sich eine schnellere Technik wie z. B. Entscheidungsbäume an.
In einem Entscheidungsbaum sind die Knoten die Kriterien für eine Gruppierung von Objekten und die Kanten die möglichen Werte dieser Kriterien.
Die meisten Merkmale der Anfragen sind deterministisch und sich nur in wenigen Clustern ähnlich. Daher können Kriterien für eine Klassifizierung gut gebildet werden. Zum Beispiel sind bei allen Anfragen innerhalb eines Clusters auf jeden Fall die Degree Sequence, die Prädikatzähler und der Join-Index-Prädikatzähler gleich. Für die Klassifizierung werden alle Cluster, die denselben Ausführungsplan haben, gruppiert. Danach werden Regeln für den Baum erstellt.
Um die Gruppierung zu beschleunigen, werden zunächst alle Planvorlagen der Clusterrepräsentanten in Postorder durchlaufen. Von diesem Durchlauf wird eine MD5 Hash Signatur erstellt. Um eine Ähnlichkeit in Plänen festzustellen, werden nun diese Signaturen anstatt der tatsächlichen Planvorlagen miteinander verglichen. Die Vorteile dieses Vorgehens sind die erhöhte Exaktheit des Klassifizierers und ein Klassifizierungsbaum von niedriger Höhe. Durch die Benutzung eines Entscheidungsbaumes kann auch dargestellt werden, welche Attribute des Merkmalsvektors den meisten Einfluss auf die Wahl eines Ausführungsplanes haben (siehe [SH04]). In Abbildung 10 ist als Beispiel ein solcher Baum zu sehen. Die Klassifizierung erfolgt hier an Hand der Effektiven Größe der Tabellen.

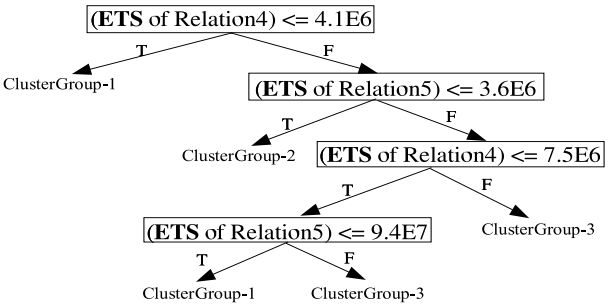

Abbildung 10: Entscheidungsbaum

Ein weiterer Vorteil von Entscheidungsbäumen ist die Tatsache, dass nach der Generierung der Regeln auf die Merkmalsvektoren verzichtet werden kann. Als Cluster werden dann die Blätter des Baumes interpretiert. In PLASTIC wurde der C5.0 decision tree induction algorithm benutzt, um die Entscheidungsbäume zu generieren (siehe [DT]).

3.5 Ausführungsplanvorlage

Als eine **Ausführungsplanvorlage** (plan template) für eine Anfrage wird ein Ausführungsplan mit allen Datenbankoperatoren bezeichnet (z. B. tablescan, sort, merge-join), bei dem alle Werte der Eingaben, auf denen diese Operatoren arbeiten, wie Tabellennamen und Attributnamen, durch Variablen ersetzt wurden (siehe [GPSH02]).
Die Planvorlage wird für die Ausführung mit den spezifischen Informationen aus der Anfrage ergänzt. Des Weiteren werden die Kosten der Operatoren mit einberechnet (siehe [SH04]). Stimmen bei zwei Anfragen die Plan-Vorlagen überein, so werden die Anfragen als zueinander gleich angesehen.

4 PLASTIC

Das Tool PLASTIC (PLAn Selection Through Incremental Clustering) (siehe [GPSH02] und [SH04]) stellt einen Ansatz zur Verbesserung von Optimierern durch mehrmalige Nutzung von einmal gemachten Ausführungplänen dar, in dem es diese speichert und bei neuen Anfragen wieder verwendet. Es kann von Optimierern benutzt werden, um Rechenzeit für die Generierung eines optimalen Ausführungsplanes zu verringen.
Der Aufbau von PLASTIC ist in dem Block-Level-Diagramm in Abbildung 11 dargestellt.
In der Beschreibung der Architektur des Tools wird durch die durchgezogenen Linien der Ablauf bei einem zutreffenden Cluster gezeigt, durch die gestrichelten Linien der Fall mit keinem übereinstimmenden Cluster.
Eine Datenbankanfrage wird zunächst im *Feature Vector Extractor* in einen Merkmalsvektor umgewandelt. Dazu sind unter anderem Informationen aus den System-Katalogen nötig. Im Modul *Similarity Check* wird auf die Gleichheit der Anfrage mit den in der *Query Cluster Database* vorhandenen Repräsentanten der Cluster verglichen. Dieser Vorgang wird durch die Benutzung eines Klassifizierers beschleunigt.

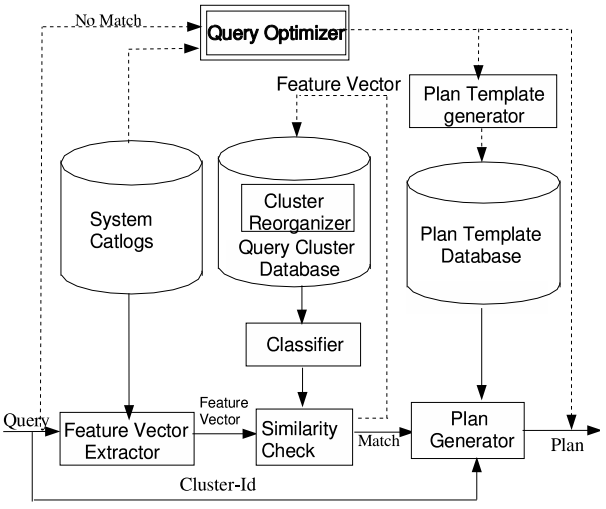

Abbildung 11: Block-Level-Diagramm

Entspricht die Anfrage einem Repräsentanten eines Clusters, so wird die Ausführungsplanvorlage des dazugehörigen Repräsentanten aus der *Plan Template Database* abgefragt. Vom *Plan Generator* wird diese in eine Anfrage umgewandelt, in dem die für die Anfrage spezifischen Eingaben in die Vorlage eingefügt werden.
Ergibt die Suche nach Ähnlichkeit jedoch keine Übereinstimmung, so wird ein üblicher Optimierer benutzt, um den besten Ausführungsplan für die Anfrage zu generieren und die Anfrage auszuführen. Gleichzeitig wird der Ausführungsplan an den *Plan Template Generator* weitergegeben, der daraus eine Planvorlage erstellt. Diese wird in der *Plan Template Database* gespeichert. Die Pläne können aus Effizienzgründen als Signaturen gespeichert werden. Der Merkmalsvektor der Anfrage wird gleichzeitig als der Repräsentant eines Clusters in der *Query Cluster Database* gespeichert.
Die Datenbank, die die Cluster abspeichert, kann durch die *Cluster Reorganisation* verändert werden, um z. B. an die Speicherplatzbedingungen angepasst zu werden. Ein anderes Beispiel wäre die Anpassung der Anzahl der Cluster durch das Entfernen von den Merkmalsvektoren und Planvorlagen der sogenannten „outlier“-Anfragen[6].

[6] Als outlier bezeichnet man im Clustering Elemente, die in keinen Cluster passen. In dem Fall von PLASTIC

Um eine tatsächliche Verbesserung eines Optimierers mit sich zu bringen, wurde das Erweiterungstool auf bestimmte Aspekte hin untersucht:

1. Die **Exaktheit** gibt an, wie oft der von PLASTIC gewählte Ausführungsplan mit dem des Optimierers übereinstimmt. Eine Differenz kommt immer dann vor, wenn ein Cluster im Merkmalsraum mehr als einen Plan im Planraum überlappt. Dieses wird durch die Größen der Cluster und die Position der Repräsentanten bestimmt. Die erwartete Wahrscheinlichkeit, dass eine Anfrage mit einem falschen Ausführungsplan assoziiert wird, ist

$$E(q) = \sum_i P_{error}\left(\frac{q}{C_i}\right) \cdot P(C_i)$$

   wobei C_i das uneindeutige Cluster,

$$P(C_i) = \frac{1}{TotalNumberOfClusters}$$

   die A-Priori-Wahrscheinlichkeit, dass eine Anfrage zu einem Cluster zugeordnet wird (alle Cluster werden als gleich wahrscheinlich angenommen), und $P_{error}(\frac{q}{C_i})$ die Wahrscheinlickeit, dass q fälschlicherweise zu einem Cluster C_i zugeordnet wird, sind. In der Regel überlappen sich zwei Pläne in einem Cluster, so dass die Wahrscheinlichkeit der fehlerhaften Zuordnung einer Anfrage zu einem Cluster auf 50 % approximiert werden kann.

2. Die **Effizienz** ist die Dauer in Zeit, die PLASTIC braucht, um eine Anfrage zu klassifizieren und den Ausführungsplan zurück zu geben. Diese Zeit wird mit der Zeit, die ein Optimierer für die Generierung eines Plans braucht, verglichen.

3. Der **Risikofaktor** ist das maximale Risiko der Planvorhersage mit PLASTIC, also die mit der Falschzuordnung einer Anfrage größte verbundene Kostensteigerung verglichen zu den Kosten des optimierer-generierten Plans.

4. Als Letztes ist der Bedarf an **Speicherplatz**, um eine Anfrage als Merkmalsvektor und Repräsentant zu speichern, zu nennen. Daraus kann der insgesamt benötigte Speicherplatz geschätzt werden. Der benötigte Speicherplatz ist proportional zum Grad der Exaktkeit.

Die Leistungsfähigkeit von PLASTIC wurde nach den oben genannten Eigenschaften an Hand des TPC-H Benchmarks (siehe [TPC]) untersucht.
Die Tabellen dieser Datenbank sind in der nebenstehenden Tabelle dargestellt.

Tabelle	Kardinalität	Größe (in MB)
PART	200.000	29,8
PARTSUPP	800.000	124,7
CUSTOMER	150.000	26,6
SUPPLIER	10.000	1,7
LINEITEM	4.859.686	658,6
NATION	25	$2 \cdot 10^{-3}$
REGION	5	$4 \cdot 10^{-4}$

Für die Arbeit wurden zwei vereinfachende Annahmen gemacht:

1. Der Status des Systems, auf dem getestet wird, ist stabil.

2. Es werden nur SPJ[7]-Anfragen behandelt, da diese mit einer beschränkten Anzahl von Operatoren arbeiten.

Des Weiteren wurde für das Testen mit dem TPC-H Benchmark angenommen, dass die Anfragen gleichmäßig im Merkmalsraum verteilt sind. Zum Vergleich wurde der Optmimierer der DB2 Universal Database Version 7 mit der Einstellung der Optimierungsklasse 5 benutzt.

sind das die gespeicherten Anfragen, zu denen mit der momentanen Anfragelast kaum Übereinstimmungen gefunden werden.

[7] SPJ heisst Selection-Projection-Join und beinhaltet im Planraum die Anfragebäume, die einer linearen Anordnung der Join-Operatoren entsprechen (siehe [CH98]).

Die Ergebnisse für das Intra-Query Plan Sharing (mit Änderung der Selektivitäten von einzelnen Tabellen in gleichbleibender Anfragevorlage) werden am Beispiel der Anfrage Q_2 in der nachfolgenden Tabelle erläutert. Für die Durchführung wurden insgesamt 65 Cluster gebildet. In der Tabelle ist P-DB2 die um PLASTIC erweiterte DB2. Hierbei gibt es noch zwei Unterteilungen: wenn der Leader-Algorithmus sowohl zum Clustering als auch zur Klassifizierung benutzt wird, und wenn die Klassifizierung durch einen Entscheidungsbaum realisiert wird.

Der große Vorteil der Arbeit mit PLASTIC liegt in der benötigten Zeit für die Klassifizierung der Anfrage und Rückgabe des Ausführungsplanes, die die Durchschnittszeit von 0,1 s der DB2 stark unterschreitet.

Metrik	DB2	P-DB2	
		Leader	Entscheidungsbaum
Exaktheit	100 %	90,76 %	88,8 %
Effizienz	0,1 s (avg. case)	$4 \cdot 10^{-3}$ s (worst case)	$2,5 \cdot 10^{-4}$ s
Speicherplatz	-	1,97 KB	3,96 KB

Im *worst case* mit Benutzung des Leader-Algorithmus, der eine brute force-Suche macht und damit die Anfrage mit allen 65 Clustern vergleichen muss, liegt die Suchzeit bei nur $4 \cdot 10^{-3}$ s.
Mit der Benutzung eines Entscheidungsbaumes zur Klassifizierung vermindert sich die benötigte Zeit noch stärker. Allerdings vergrößert sich die Häufigkeit von Falschzuordnungen, was mit dem decision-tree-Algorithmus zusammen hängt.

Die Exaktheit von 100 % der DB2 resultiert aus der Annahme, dass sie tatsächlich immer den besten Plan bei Level 5 findet. Die Leistung der Datenbank in der Realität ist jedoch nicht so gut wie im höchsten Level 9, in welchem aus Kosten des Rechenaufwandes nicht immer gearbeitet wird. Auf Grund der Tatsache, dass DB2 auf Level 5 lief, konnte von PLASTIC sogar ein besserer Plan mit nierdrigeren Kosten als der Plan des Optimierers gefunden werden (siehe letzte Zeile der obigen Tabelle). In dieser Tabelle sind die Kosten in Abhängigkeit von der unterschiedlichen Anzahl mehrdeutiger Cluster[8] dargestellt.

Anzahl uneindeutiger Cluster	Kosten von DB2	Kosten von P-DB2	Risikofaktor (in %)
1	261.209	266.260	1,9
2	241.054	246.000	2
3	173.913	188.684	1,1
4	158.577	158.681	0
5	161.814	159.078	-0,02

Aus der Tabelle erkennt man, dass das Risiko einer anderen Entscheidung als DB2 ≤ 2 % ist[9]. Dieses ist darauf zurückzuführen, dass PLASTIC in den Fehlentscheidungen meistens den zweitbesten Ausführungsplan wählt, so dass die dadurch entstehenden Mehrkosten sehr gering sind. Solche Fehlentscheidungen passieren dann, wenn die Änderung des Ausführungsplanes sich in einem Cluster befindet. An den Änderungepunkten sind die Kosten der Pläne zueinander sehr ähnlich, so dass der Unterschied sehr gering ist.
In Tests zum inter-query-Plan-sharing konnten ähnlich gute Ergebnisse erzielt werden. Dabei wurden sowohl Projektionsattribute, Selektionsprädikate, Join Attribute und ganze Tabellen von Anfragen aus dem Benchmark verändert.
PLASTIC kann sowohl offline mit künstlichen Anfragen zur Erstellung von vorgefertigten Clustern als auch online in Bezug auf Clusterbildung und Planauswahl arbeiten. In diesem Fall wird der Anfrageplan inkrementell mit dem Eintreffen von neuen Anfragen erschlossen.
Die von den Datenbanken bekannten „plan hints“ zur Beeinflussung des Optimierers werden in der Anwendung auf das ganze Cluster ausgeweitet und passen sich so den Einstellungen des Benutzers an.
Eine in 2004 vorgestellte Erweiterung war die Berechnung und Einbeziehung von Kostenabschätzungen in die Plangenerierung, ein automatisches Plan-Kosten-Diagramm, welches die verschiedenen vom Optimierer gewählten Pläne für einen Anfrageraum mit den dazugehörigen Kosten darstellt, sowie ein Modul zum Vergleichen von Ausführungsplänen innerhalb und zwischen verschiedenen Datenbankplattformen. Die Kosten für einen Ausführungsplan können nicht aus den Daten des Repräsentanten übernommen werden, da die Anfragen zwar denselben Plan teilen, jedoch sehr unterschiedlich sein können.

[8] Cluster, die mehr als einen Plan im Planraum abedecken
[9] Die Kostendarstellung ist in *timerons*, der DB2-Kosteneinheit.

Daher werden die Kosten des Repräsentanten im *Plan Generator* durch ein Schätzungs-Modul skaliert und interpoliert, um an die neue Anfrage angepasst zu werden. Diese Informationen werden in den neuen Ausführungsplan mit einbezogen (siehe [SH04]). In Abbildung 12 ist ein Teil des Moduls zur Plananalyse dargestellt, welcher zwei Ausführungspläne im selben Anfrageraum aufzeigt. Der Unterschied in den Plänen liegt im Zugriff auf die PARTSUPP-Relation: durch einen Index Scan und einen Table Scan.

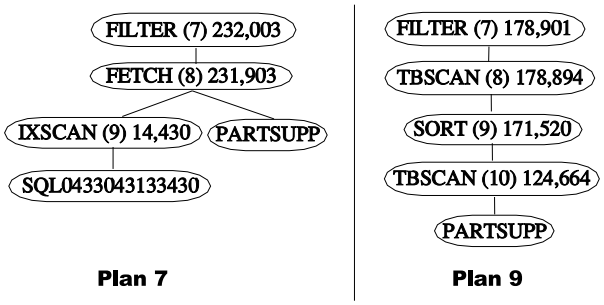

Abbildung 12: Vergleich von Plänen

5 Zusammenfassung und Ausblick

Die in diesem Seminarbeitrag vorgestellten Methoden dienen in Kombination dazu, die Kosten für die Optimierung von Ausführungsplänen zu verringern. Es wird versucht, effizient vorauszusagen, welchen Plan der Optimierer für eine Anfrage gewählt hätte. Dieses ist abhängig von der Anzahl der schon getätigten Anfragen an die Datenbank, da die Cluster erst inkrementell gebildet werden müssen. Eine Alternative stellt ein passendes Trainingset dar, mit welchem Cluster im Voraus gebildet werden.
Die Versuche mit dem TPC-H Benchmark zeigten sehr gute Voraussagen in den Ausführungsplänen, verbunden mit einer um ein Vielfaches erhöhten Optimierungszeit der Anfragen. Die Erhöhung der Kosten bei falschen Voraussagen der Plänen ist sehr gering, der durch das Tool benötigte Speicherplatzbedarf ist niedrig.
Die Kombination von einem Optimierer und dem Tool bringt eine erhebliche Erhöhung der Leistung nicht nur in der Geschwindigkeit, sondern auch in der Möglichkeit für den Optimierer immer im höchsten Optimierungslevel zu arbeiten, mit sich. Da die Berechnungen des Optimierers einmalig sind, verteilen sich die Mehrkosten für den höheren Optimerungslevel auf alle Anfragen, deren Pläne durch das Clustering generiert werden.
Die zukünftige Arbeit am Tool PLASTIC wird sich auf die Erweiterung von SPJ-Anfragen auf verschachtelte Anfragen, Gruppierungen und Aggregierungen richten. Des Weiteren sollen die Gewichtungen und der Schwellwert für den Vergleichsalgorithmus automatisch berechnet und gesetzt werden können.

Literatur

[CH98] **Chaudhuri, S.;** An Overview of Query Optimization in Relational Databases, Proc. of 17. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Juni 1998

[DBS05] **Lipeck, U., Makoui, M.:** Vorlesung „Grundlagen der Datenbanksysteme (Datenbanksysteme I)“, WS2004/05, Universität Hannover

[DT] http://www.rulequest.com/see5-info.html

[EID00] **Eidenberger, H.:** Suchmodellbasiertes Content-based Image Retrieval, Dissertation, Sozial- und wirtschaftswissenschaftliche Fakultät, Universität Wien, Juli 2000

[GPSH02] **Ghosh, A., Parikh J., Sengar V., Haritsa J.:** Plan Selection based on Query Clustering, Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB), August 2002.

[H03] **Haritsa, J.** Reducing Query Optimization Overheads through Plan Recycling, Power Point Präsentation, September 2003

[H05] **Haney, F.** DOAG-Vortrag 2005
http://www.doag.org/pub/docs/regio/jena/2005-05/Einleitung_Dr._Haney.pdf

[MR05] **Mishne, G., de Rijke, M.:** Vorlesung „Internet Information“, SS 2005, University of Amsterdam
http://lit.science.uva.nl/Teaching/0405/II/page9.html

[SAR04] **Sarda, P.:** Green Query Optimization: Taming query optimization overheads through plan recycling, a project report submitted in partial fulfilment of the requirements for the Degree of Master of Engineering in Computer Science and Engineering, Department of Computer Science and Automation, Indian Institute of Science, Juli 2004.

[SEL+79] **Griffiths Selinger, P., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., Price, T. G.:** Access Path Selection in a Relational Database Management System, Proc. ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, Mai 1979.

[SH04] **Sarda, P., Haritsa, J.R.:** Green Query Optimization: Taming Query Optimization Overheads through Plan Recycling, Proc. of 30th Intl. Conf. on Very Large Data Bases (VLDB), August 2004.

[SQL] http://www.sql-server-performance.com/sql_server_performance_audit8.asp

[STAT] http://www.statsoftinc.com/textbook/stcluan.html#h

[TPC] http://www.tpc.org

[WDM04] **Henze, N.:** Vorlesung „Web Data Management“, WS 2004/05, Universität Hannover