Minimale Überdeckung

Eine Menge funktionaler Abhängigkeiten $\mathcal{F}^{min}$ ist eine *minimale Überdeckung* zu $\mathcal{F}$, wenn wir sie durch Anwendung der folgenden Schritte erzeugen können:

- Führe alle möglichen Linksreduktionen durch.
- Führe alle möglichen Rechtsreduktionen durch.
- Streiche alle trivialen funktionalen Abhängigkeiten der Form $X \to \emptyset$.
- Vereinige alle funktionalen Abhängigkeiten mit gleicher linker Seite $X \to Y_1, \ldots, X \to Y_n$ zu einer einzigen FA der Form $X \to Y_1 \ldots Y_n$[1].

[1] Durch Anwendung von (A4).

Entscheidung mittels XPlus-Algorithmus

- Sei $X \to Y$ eine Abhängigkeit in $\mathcal{F}$ und sei $Z \to Y$, wobei $Z \subseteq X$.
  Wir führen die entsprechende Linksreduktion durch, wenn $XPlus(Z, Y, \mathcal{F})$ das Ergebnis `true` liefert.
- Sei $X \to Y$ eine Abhängigkeit in $\mathcal{F}$ und sei $X \to Z$, wobei $Z \subseteq Y$.
  Wir führen die entsprechende Rechtsreduktion durch, wenn $XPlus(X, Y, \mathcal{F}')$[2] das Ergebnis `true` liefert.

Satz

Sei eine Menge funktionaler Abhängigkeiten $\mathcal{F}$ gegeben und sei $\mathcal{F}'$ aus $\mathcal{F}$ durch eine Links- oder Rechtsreduktion hervorgegangen.

Dann $\mathcal{F} \equiv \mathcal{F}'$.

[2] $\mathcal{F}' = (\mathcal{F} \setminus \{X \to Y\}) \cup \{X \to Z\}$

3. Normalform

Ein Relationsschema $R = (V, \mathcal{F})$ ist in *3. Normalform* (3NF) genau dann, wenn jedes NSA $A \in V$ die folgende Bedingung erfüllt.

Wenn $X \to A \in \mathcal{F}$, $A \notin X$, dann ist X ein Superschlüssel.

Boyce-Codd-Normalform

Ein Relationsschema $R = (V, \mathcal{F})$ ist in *Boyce-Codd-Normalform* (BCNF) genau dann, wenn die folgende Bedingung erfüllt ist.

Wenn $X \to A \in \mathcal{F}$, $A \notin X$, dann ist X ein Superschlüssel.

Übungsblatt Nr. 11

Übung Nr. 2

Betrachten Sie Mengen von Attributen V, $X \subseteq V$ und ein Menge von funktionalen Abhängigkeiten $\mathcal{F}$ wie folgt:

$$V = \{A_1, \ldots, A_n, B_1, \ldots B_n, C_1, \ldots, C_n, D\},$$
$$\mathcal{F} = \{A_i \to C_i, B_i \to C_i \mid 1 \leq i \leq n\} \cup \{C_1 \ldots C_n \to D\},$$
$$X = A_1 \ldots A_n B_1 \ldots B_n D.$$

(1) Zeigen Sie, dass $\mathcal{F} = \mathcal{F}^{min}$.

(2) Zeigen Sie die Gültigkeit der folgenden Aussagen:
- $A_1 A_2 \ldots A_n \to D \in \mathcal{F}^+$,
- $B_1 B_2 \ldots B_n \to D \in \mathcal{F}^+$,
- Sei $w \in \{A_1, B_1\} \times \ldots \times \{A_n, B_n\}$. $w \to D \in \mathcal{F}^+$.

(3) $\mathcal{F}$ enhält gerade gerade $2n + 1$ funktionale Abhängigkeiten. Zeigen Sie, dass $\pi[X]\mathcal{F}$ mindestens 2^n Elemente hat.

Übung Nr. 2 (1)

Zu zeigen: $\mathcal{F} = \mathcal{F}^{min}$, d.h. es sind *keine* Links- und Rechtsreduktionen möglich.

(a) Linksreduktionen: Möglich, wenn $XPlus(Z, Y, \mathcal{F}) = \texttt{true}$, wobei $X \to Y \in \mathcal{F}$ und $Z \to Y$ mit $Z \subseteq X$.
- Somit sind die FDs $A_i \to C_i$ und $B_i \to C_i$, $1 \leq i \leq n$ offenkundig *nicht* linksreduzierbar.
- Betrachte FD $C_1 \ldots C_n \to D$. Diese kann auch nicht linksreduziert werden, da sie die einzige funktionale Abhängigkeit ist, die das Attribut D auf der rechten Seite hat und es auch keine Möglichkeit gibt, ein beliebig entnommenes C_i über die anderen funktionalen Abhängigkeiten herzuleiten.
- *Keine* Linksreduktion möglich!

Übung Nr. 2 (1)

Zu zeigen: $\mathcal{F} = \mathcal{F}^{min}$, d.h. es sind *keine* Links- und Rechtsreduktionen möglich.

(a) Rechtsreduktionen: Möglich, wenn $XPlus(X, Y, \mathcal{F}') = \texttt{true}$, wobei $X \to Y \in \mathcal{F}$ und $X \to Z$ mit $Z \subseteq Y$.
$\Rightarrow \mathcal{F}' = (\mathcal{F} \setminus \{X \to Y\}) \cup \{X \to Z\}$
- Somit ist die FD $C_1 \ldots C_n \to D$ offenkundig *nicht* linksreduzierbar.
- Betrachte FDs $A_i \to C_i$ und $B_i \to C_i$, $1 \leq i \leq n$ parallel. Reduziere o.B.d.A. eine beliebige Abhängigkeit $A_i \to C_i$ zu $A_i \to \emptyset$. Dafür müsste mit $\mathcal{F}' = (F \setminus \{A_i \to C_i\}) \cup \{A_i \to \emptyset\})$ $XPlus(A_i, C_i, \mathcal{F}') = \texttt{true}$ als Resultat liefern. Dies ist nicht der Fall, da $A_i^+ = A_i$ mit $\mathcal{F}'$.
- *Keine* Rechtsreduktion möglich!

- Es gibt keine funktionalen Abhängigkeiten mit gleicher linker Seite. Somit ist keine weitere Vereinfachung von $\mathcal{F}$ möglich und es gilt somit $\mathcal{F} = \mathcal{F}^{min}$.

Übung Nr. 2 (2)

Zu zeigen: $A_1A_2...A_n \to D \in \mathcal{F}^+$, d.h., dass $XPlus(A_1A_2 \ldots A_n, D, \mathcal{F}) = \texttt{true}$.
Betrachte hierfür die Anwendung des *XPlus*-Algorithmus:

- $\texttt{result} = \{A_1A_2 \ldots A_n\}$
- Runde 1: $A_1 \to C_1 \Rightarrow \texttt{result} := \texttt{result} \cup C_1$
- ...
- Runde n: $A_n \to C_n \Rightarrow \texttt{result} := \texttt{result} \cup C_n$
- Runde $n+1$: $C_1 \ldots C_n \to D \Rightarrow \texttt{result} := \texttt{result} \cup D$
- Keine Veränderung von `result` in Runde $n+2$
- $\Rightarrow$ Da $XPlus(A_1A_2 \ldots A_n, D, \mathcal{F}) = \texttt{true}$ gilt $A_1A_2 \ldots A_n \to D$.

Übung Nr. 2 (2)

- **Zu zeigen:** $B_1B_2 \ldots B_n \to D \in \mathcal{F}^+$.
$\Rightarrow$ Lösung analog zu $A_1A_2 \ldots A_n \to D$.
- **Zu zeigen:** $w \to D \in \mathcal{F}^+$, wobei $w \in \{A_1, B_1\} \times \ldots \times \{A_n, B_n\}$.
$\Rightarrow$ Lösung analog zu $A_1A_2 \ldots A_n \to D$, wobei entweder die FD $A_i \to C_i$ oder $B_i \to C_i$ verwendet wird, abhängig davon, ob $w_i = A_i$ oder $w_i = B_i$.

Übung Nr. 2 (3)

Zu zeigen: $\pi[X]\mathcal{F}$ hat mindestens 2^n Elemente, wobei $X = A_1 \dots A_n B_1 \dots B_n D$.

- Für $Z \subseteq V$ ist $\pi[Z]\mathcal{F} := \{X \to Y \in \mathcal{F}^+ \mid XY \subseteq Z\}$.
- Da oben schon gezeigt wurde, dass $w \to D \in \mathcal{F}^+$ gilt, mit $w \in \{A_1, B_1\} \times \dots \times \{A_n, B_n\}$ und nach Definition alle diese w in $\pi[Z]\mathcal{F}$ enthalten sind ($w \to D$ ist genau definiert über $X = \{A_1 \dots A_n B_1 \dots B_n D\}$), ist es ausreichend zu zeigen, dass es mindestens 2^n "Varianten" von w gibt.
- Dies ist der Fall, was sich einfach zeigen lässt: es gibt n-mal die Möglichkeit sich für eine der Attribute A_i oder B_i zu entscheiden, also 2^n Möglichkeiten ein w zu wählen.

Übungsblatt Nr. 11

Übung Nr. 3

Sei ein Schema R(A,B,C) gegeben.

(a) Geben Sie ein Menge $\mathcal{F}_1$ von funktionalen Abhängigkeiten so an, dass R nicht in 3NF.

(b) Geben Sie bzgl. $\mathcal{F}_1$ eine verlustfreie und abhängigkeitsbewahrende Zerlegung in 3NF an.

(c) Geben Sie eine Menge $\mathcal{F}_2$ von funktionalen Abhängigkeiten so an, dass R in 3NF, aber nicht in BCNF.

(d) Geben Sie bzgl. $\mathcal{F}_2$ eine verlustfreie Zerlegung in BCNF an.

(e) Begründen Sie, warum es bzgl. $\mathcal{F}_2$ keine verlustfreie *und* abhängigkeitsbewahrende Zerlegung in BCNF geben kann.

Übung Nr. 3

(a) $\mathcal{F}_1 = \{A \to B, B \to C\}$

(b) $\rho = \{(A, B), (B, C)\}$

(c) $\mathcal{F}_2 = \{A\,B \to C, C \to B\}$

(d) $\rho = \{(A, C), (C, B)\}$

(e) $\pi[AC]\mathcal{F}_2$ enthält nur triviale FAs; $\pi[CB]\mathcal{F}_2$ enthält als einzige nicht-triviale FA $C \to B$. Um die Abhängigkeiten in $\mathcal{F}_2$ wiederherzustellen, müßte man B wieder zu (A,C) hinzunehmen, was aber der geforderten Zerlegung nach BCNF widerspricht.

Rückblick

Berechnung des θ-Verbund

Seien ein Relationsschemata $R(A, B)$ und $S(C, D)$ mit Instanzen r und s gegeben, wobei gilt:

- r benötigt N und s benötigt M Seiten der Größe K,
- die Tupel in r und s haben gleiche Länge und eine Seite kann maximal k solcher Tupel aufnehmen.

$\Rightarrow$ Es soll der θ-Verbund $R \bowtie_{B=C} S$ berechnet werden.

Frage: Wieviele Seiten benötigt das Ergebnis der Verbundberechnung mindestens und höchstens[3]?

[3] Jede Seite soll jeweils maximal viele Tupel des Ergebnisses aufnehmen.

Berechnung des θ-Verbund

- **Minimale Anzahl:** 0 (leere Menge)
- **Maximale Anzahl:** Tupelgröße in R und S: $\frac{K}{k}$, Tupelgröße im Ergebnis: $2 \cdot \frac{K}{k}$ (*alle* Attribute). Maximale Anzahl Tupel in S: $N \cdot k$, in R: $M \cdot k$, folglich maximal

$$k^2 \cdot N \cdot M$$

Tupel im Kreuzprodukt. Die Ergebnisgröße ist also:

$$2\frac{K}{k} \cdot k^2 \cdot N \cdot M = K \cdot 2k \cdot NM$$

Dafür braucht man $2k \cdot NM$ Seiten.

8 Auswertung von Anfrageoperatoren
8.1 Selektion

Auswertung von $\sigma[A\ op\ val]R$.

Sofern vorhanden, kann

- Index zu A,
- Sortierung zu A

ausgenutzt werden. Ansonsten wird ein Scan über R erforderlich.

8.2 Projektion

Auswertung von $\pi[A_1, \ldots, A_m]R$.

Scan über R – anschließend müssen gegebenenfalls Duplikate entfernt werden mittels

- Sortierung, oder
- eines Hashverfahrens.

8.3 Verbund

Auswertung von $R \bowtie S$.

Sei X das Format von R und Y das Format von S. Sei n die Anzahl Tupel der betrachteten Relation r zu R und m entsprechend die Anzahl Tupel von s zu S.

- *Nested-Loop-Verbund*,
- *Sort-Merge-Verbund*,
- *Hash-Verbund*.

Nested-Loop-Verbund

```
FOR EACH tuple μ ∈ r DO
    FOR EACH tuple ν ∈ s DO
        IF μ[X ∩ Y] = ν[X ∩ Y] THEN add μ ⋈ ν to result
```

Variante Block-Nested-Loop-Verbund

Angenommen der gesamte Datenbankpuffer bestehend aus B Seiten steht zur Verfügung.

```
FOR EACH block of B − 2 pages of r DO
    FOR EACH page of s DO
        for all matching current in-memory tuples μ of the r-block and ν the current s-page
add μ ⋈ ν to result.
```

Sei N die Anzahl Blöcke von r und M die Anzahl Blöcke von s.

- Sei o.B.d.A. $N \leq M$. Wähle r als die äußere Relation.
- Sei r die äußere Relation. Die Anzahl der gelesenen Blöcke ergibt sich zu $N + N \times M$.

Nested-Loop-Verbund

```
FOR EACH tuple μ ∈ r DO
    FOR EACH tuple ν ∈ s DO
        IF μ[X ∩ Y] = ν[X ∩ Y] THEN add μ ⋈ ν to result
```

Variante Index-Nested-Loop-Verbund

- Existiert ein Index über die Verbundattribute einer Relation, dann wähle diese als die innere Relation.

  Für jedes μ der äußeren Relation können die potentiellen Verbundpartner ν der inneren Relation mittels Index-Look-Up bestimmt werden.
- Ein Index für die innere Relation kann auch während des ersten Durchlaufs dynamisch berechnet werden: *index-on-the-fly*.

Sort-Merge-Verbund

Beide Relationen sind nach den Verbundattributen aufsteigend sortiert.

- Für jede Relation wird ein Zeiger definiert, der zu Beginn jeweils das erste Tupel referenziert.
- Erfüllen die referenzierten Tupel die Verbundbedingung, dann führe für sie den Verbund aus.
- Anderenfalls verschiebe den Zeiger, der das Tupel mit kleinerem Verbundwert referenziert, bis beide Zeiger ein weiteres Paar von Verbundpartnern referenzieren, oder der gerade bewegte Zeiger ein Tupel mit einem größeren Wert referenziert.
- In diesem Fall wird das Verfahren fortgesetzt indem letzterer Zeiger in entsprechender Weise weiter geschoben wird.

Sei N die Anzahl Blöcke von r und M die Anzahl Blöcke von s.

- Sind die Verbundattribute in beiden Relationen Schlüssel, dann werden $N + M$ Blöcke gelesen.
- Sind die Verbundattribute in beiden Relationen keine Schlüssel, dann werden im worst-case $N \times M$ Blöcke gelesen.

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
$\rightarrow$	**1**		**1**	$\leftarrow$
	2		4	
	3		7	
	4			
	5			

$\Rightarrow R \bowtie S = \{\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
$\rightarrow$	**1**		**1**	$\leftarrow$
	2		4	
	3		7	
	4			
	5			

$\Rightarrow R \bowtie S = \{\mathbf{1}\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
	1		1	
$\rightarrow$	**2**		**4**	$\leftarrow$
	3		7	
	4			
	5			

$\Rightarrow R \bowtie S = \{1\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
$\rightarrow$	**1**		1	
	2		**4**	$\leftarrow$
	3		7	
	4			
	5			

$\Rightarrow R \bowtie S = \{1\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
	1		1	
	2		**4**	$\leftarrow$
$\rightarrow$	**3**		7	
	4			
	5			

$\Rightarrow R \bowtie S = \{1\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
	1		1	
	2		**4**	$\leftarrow$
	3		7	
$\rightarrow$	**4**			
	5			

$\Rightarrow R \bowtie S = \{1\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
	1		1	
	2		**4**	$\leftarrow$
	3		7	
$\rightarrow$	**4**			
	5			

$\Rightarrow R \bowtie S = \{1, \mathbf{4}\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
	1		1	
	2		4	
	3		**7**	$\leftarrow$
$\rightarrow$	**4**			
	5			

$\Rightarrow R \bowtie S = \{1, 4\}$

Sort-Merge-Verbund: $R \bowtie S$

	R	$\bowtie$	S	
	1		1	
	2		4	
	3		**7**	$\leftarrow$
	4			
$\rightarrow$	**5**			

$\Rightarrow R \bowtie S = \{1, 4\}$

Hash-Verbund

Idee

Nur solche Tupel $\mu \in r$ und $\nu \in s$ können Partner bei Berechnung eines natürlichen Verbundes sein, für die für eine gegebene Hashfunktion h gerade $h(\mu) = h(\nu)$.

Verfahren

Sei N die Anzahl Seiten von r und sei M die Anzahl Seiten von s, wobei $N \leq M$. Die Relationen werden in Form von Blöcken gespeichert. Ein Block besteht aus einer Menge von Seiten. Die Anzahl der zur Verbundberechnung im Hauptspeicher zur Verfügung stehenden Seiten sei k.

Sei h_1 eine Hashfunktion, die jedem Tupel einer Relation einen Block zuordnet. Sei h_2 eine von h_1 verschiedene Hashfunktion, die jedem Tupel einer Relation einen Hauptspeicherbereich zuordnet.

(1) Wende h_1 jeweils auf r und s an und bilde so je eine Partition von r und s.

(2) Bilde dann den Verbund,

(2i) indem ein Block der Partition von r gelesen wird um die Tupel in den Seiten des eingelesenen Blockes mittels h_2 im Hauptspeicher zu verteilen,

(2ii) und dann die einzelnen Seiten des gemäß h_1 korrespondierenden Blockes der Partition von s nacheinander gelesen werden und für jedes Tupel einer solchen Seite die korrespondierenden Tupel von r mittels h_2 bestimmt werden.

(3) Iteriere das Verfahren bis alle Blöcke der Partition von r bearbeitet.

8.4 Mengenoperatoren und Aggregierung

$\cap$, $\cup$ und $-$

Gleichheit der beteiligten Formate!

- $r \cap s = r \bowtie s$.
- $\cup$ benötigt Duplikateliminierung.
  - $-$ ist eine Variante einer Duplikateliminierung.

`GROUP BY`, `SUM`, `AVG`, `MAX`, `MIN` und `COUNT`

- Sortierung nach den Gruppierungsattributen,

  Scan der einzelnen Gruppen.
- Existiert eine Indexstruktur zu der gegebenen Relation, dann können u.U. anstatt der eigentlichen Tupel der Relationen die entsprechenden Suchschlüsselwerte verarbeitet werden.
  - Enthält der Suchschlüssel des Index alle für die Aggregierung benötigten Attribute, dann erübrigen sich Zugriffe zu den eigentlichen Tupeln.
  - Bilden die Gruppierungsattribute einen Prefix des Suchschlüssels, dann wird der initiale Sortierschritt überflüssig.

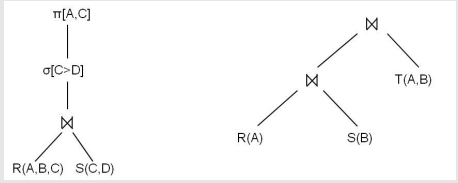
8.5 Optimierung

Optimierungstechniken

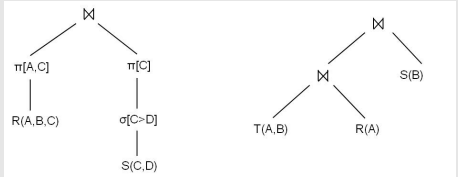
- *semantische*: Ausnutzen von Informationen über Integritätsbedingungen und (funktionalen) Abhängigkeiten,
- *algebraische*: äquivalenzbewahrende Umformungen,
- *physische*: Auswählen von Zugriffspfaden (Indexstrukturen).

Algebraische Optimierung

Anfragebäume zweier Algebraausdrücke

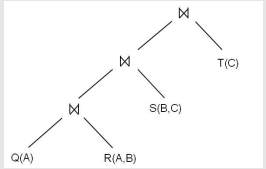


äquivalenzerhaltende Umformungen obiger Ausdrücke

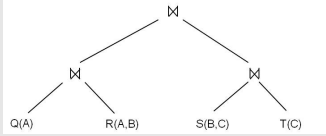


Aufgrund der Gültigkeit des Assoziativgesetzes für den Verbund ergeben sich vielfältige Möglichkeiten der Auswertung eines Verbundausdrucks mit $n > 2$ Relationen.

left-deep-tree Join-Auswertung



bushy-tree Join-Auswertung



Bei einer *left-deep-tree* Join-Auswertung ist im Unterschied zur *bushy-tree* Join-Auswertung ein Verbund-Partner jeweils immer bereits berechnet.

Physische Optimierung

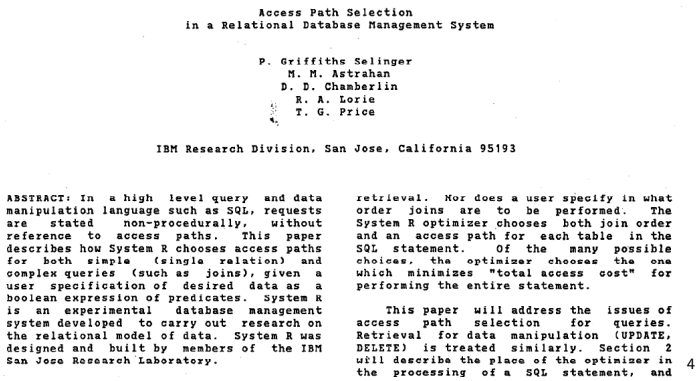
- Grundlage für Auswahlentscheidungen der physischen Optimierung sind Informationen über die Relationen und ihre Indexstrukturen.
- Diese Informationen werden im *Katalog* gehalten.
- Für die Entscheidung, ob ein Index für die Auswertung einer Anfrage von Nutzen ist, ist die *Selektivität* des Index eine wichtige Information.
  Die Selektivität ist umso stärker, je kleiner ihr Wert ist.
  - $\sigma[\alpha](R)$: $sel_\alpha = \frac{|\sigma[\alpha](R)|}{|R|}$.
    Spezialfall $\alpha \equiv A = c$ und A Schlüssel von R?
    $sel_\alpha = \frac{1}{|R|}$.
  - $R \bowtie S$: $sel_{RS} := \frac{|R \bowtie S|}{|R \times S|} = \frac{|R \bowtie S|}{|R| \cdot |S|}$.
    Spezialfall $R \bowtie_{R.A=S.B} S$ und A Schlüssel?
    $sel_{RS} \leq \frac{1}{|R|}$ wegen $| R \bowtie_{R.A=S.B} S | \leq |S|$.

left-deep-tree vs. *bushy-tree* Join-Auswertung

- Bei einer *left-deep-tree* Join-Auswertung, z.B. $(R \bowtie S) \bowtie T$, kann auf eine vollständige Materialisierung der Zwischenergebnisse von $R \bowtie S$ verzichtet werden; Zwischenresultate werden direkt an den Folgeoperator weitergereicht, hier $\bowtie T$.

  Diese Technik ist allgemeiner anwendbar, z.B. auch bei $\sigma[A = 5 \wedge B > 6]R$, wobei je ein Index über A und B angenommen wird; man redet von *Pipelining*.
- Bei einer *bushy-tree* Join-Auswertung, z.B. $(Q \bowtie R) \bowtie (S \bowtie T)$ müssen in der Regel beide Zwischenergebnisse zuerst berechnet werden, bevor der Verbund berechnet werden kann; das Berechnen der Zwischenergebnisse zu $(Q \bowtie R)$ und $(S \bowtie T)$ kann jedoch *parallel* erfolgen.

8.6 Empfohlene Lektüre



Access Path Selection
in a Relational Database Management System

P. Griffiths Selinger
M. M. Astrahan
D. D. Chamberlin
R. A. Lorie
T. G. Price

IBM Research Division, San Jose, California 95193

ABSTRACT: In a high level query and data manipulation language such as SQL, requests are stated non-procedurally, without reference to access paths. This paper describes how System R chooses access paths for both simple (single relation) and complex queries (such as joins), given a user specification of desired data as a boolean expression of predicates. System R is an experimental database management system developed to carry out research on the relational model of data. System R was designed and built by members of the IBM San Jose Research Laboratory.

retrieval. Nor does a user specify in what order joins are to be performed. The System R optimizer chooses both join order and an access path for each table in the SQL statement. Of the many possible choices, the optimizer chooses the one which minimizes "total access cost" for performing the entire statement.

This paper will address the issues of access path selection for queries. Retrieval for data manipulation (UPDATE, DELETE) is treated similarly. Section 2 will describe the place of the optimizer in the processing of a SQL statement, and

[4]

[4] In: ACM SIGMOD Conference, 1979. Kann gegoogelt werden.

Ausblick

Hashing

Betrachten Sie folgende Einträge der Tabelle *Airport*(<u>*Code*</u>, *City*):

Code	City
AAA	Anaa
...	...
ZZV	Zanesville

Geben Sie eine Hash-Funktion $h(K) = f(K)\ mod\ M$ an, die auf dem Primärschlüssel angewendet werden soll.

(i) Wählen Sie hierzu eine entsprechende Darstellung des Schlüssels $f(K)$ und das kleinstmögliche M, welches sicherstellt, dass es auch bei vollständiger Belegung des Schlüsselraums zu *keinen* Kollisionen kommen kann.

(ii) Wenden Sie Ihre Hash-Funktion auf die obigen beiden Einträge "AAA" und "ZZV" an.