Kapitel 7: Auswertung von Anfrageoperatoren
7.1 Operatoren über einer Relation

Selektion: $\sigma[A\ op\ val]R$

Sofern vorhanden, kann

- Index zu A,
- Sortierung zu A

ausgenutzt werden. Ansonsten wird ein Scan über R erforderlich.

Projektion: $\pi[A_1, \ldots, A_m]R$

Scan über R – anschließend müssen gegebenenfalls Duplikate entfernt werden mittels:

- Sortierung

  oder

- Hashverfahren

7.2 Mengenoperatoren und Aggregierungen

Mengenoperatoren: $\cap$, $\cup$ und $-$

Gleichheit der beteiligten Formate!

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

Aggregierung: `GROUP BY` mit `SUM`, `AVG`, `MAX`, `MIN` und `COUNT`

- Sortierung nach Gruppierungsattributen, Scan der einzelnen Gruppen.
- Existiert eine Indexstruktur zu der gegebenen Relation, dann sind u.U. Optimierungen möglich:
  - Bilden die Gruppierungsattribute einen Prefix des Suchschlüssels oder sind identisch mit ihm, dann wird der initiale Sortierschritt überflüssig.
  - Enthält der Suchschlüssel des Index sogar alle benötigten Attribute, dann erübrigen sich Zugriffe zu den eigentlichen Tupeln.

7.3 Verbundauswertung

Verbund: $R \bowtie S$

Sei X das Format von R und Y das Format von S, d.h. $R(X)$ und $S(Y)$.
Sei n die Anzahl Tupel der betrachteten Relation r zu R
und m entsprechend die Anzahl Tupel der betrachteten Relation s zu S.

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

Nested-Loop-Verbund

Verfahren

`FOR EACH` tuple $\mu \in r$ `DO`
    `FOR EACH` tuple $\nu \in s$ `DO`
        `IF` $\mu[X \cap Y] = \nu[X \cap Y]$ `THEN` add $\mu \bowtie \nu$ to result

Variante 1: 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 current matching in-memory tuples μ of the r-block
        and ν of the current s-page add $\mu \bowtie \nu$ to result.

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

- Sei o.B.d.A. $N \leq M$. Dann wähle r als die äußere Relation.
- Sei r die äußere Relation. Die Anzahl der gelesenen Seiten ergibt sich zu: $N + \lceil N/(B-2) \rceil \times M$

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

Variante 2: Index-Nested-Loop-Verbund

- Existiert ein Index über den Verbundattributen 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

Voraussetzung: Beide Relationen sind nach den Verbundattributen aufsteigend sortiert.

Verfahren unter *vereinfachender* Annahme, dass die Werte der Verbundattribute jeweils eindeutig sind.

- 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 und verschiebe einen der Zeiger weiter.
- 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 der zuletzt nicht bewegte Zeiger in entsprechender Weise verschoben wird.

Seitenzugriffe

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

- *Untere Schranke*: Es werden mindestens $M + 1$ Seiten gelesen.

  Sofern Suchschlüsselwert des ersten Tupels von r größer als alle Suchschlüsselwerte in s.

- *Obere Schranke*: Es werden höchstens $N \times M$ Seiten gelesen.

  Sofern in r und s nur ein und derselbe Suchschlüsselwert existiert.

- *Vereinfachende Annahme*: Die Werte der Verbundattribute sind jeweils eindeutig.

  Mittlere Seitenzugriffe: Unter dieser Annahme werden typischerweise $N + M$ Seiten gelesen.

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 (Hash-Partitioned-Join, Grace Hash Join)

Sei h_1 eine Hashfunktion, die jedem Tupel einer Relation eine Partition 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 Partitionierung von r und s.

(2) Bilde dann den Verbund:

- (2i) Lese eine Partition der kleineren Relation r und verteile die Tupel in den Seiten mittels h_2 im Hauptspeicher (*build*).
- (2ii) Lese dann die gemäß h_1 korrespondierende Partition von s und bestimme die passenden Tupel mittels h_2 (*probe*).

(3) Iteriere das Verfahren bis alle Partitionen von r bearbeitet.

Seitenzugriffe

Sei N die Anzahl Seiten von r und sei M die Anzahl Seiten von s, wobei $N \leq M$.
Die Anzahl der zur Verfügung stehenden Seiten im Hauptspeicher sei k.

- Phase 1 (Partitionierungsphase): $2(M + N)$
- Phase 2: $M + N$
- Insgesamt: $3(M + N)$

Unter Annahme, dass für die Partitionierung genügend Interspeicher zur Verfügung steht (Anzahl Partitionen $\leq k - 1$) und eine Partition der kleineren Relation nicht mehr als $k - 2$ Seiten benötigt.

7.4 Verbund-Index

Idee

Vorberechne potenzielle Verbundoperationen und speichere das Resultat in einer Tabelle mit Spalten gerade die Primärschlüsselwerte/TIDs der Verbundpartner.

Beispiel

`Customer`

cnr	cname	city	age	job
1	Smith	Boston	21	clerk
2	Collis	Austin	26	secretary
3	Ross	Austin	36	manager
4	Jones	Paris	29	engineer

`Sales`

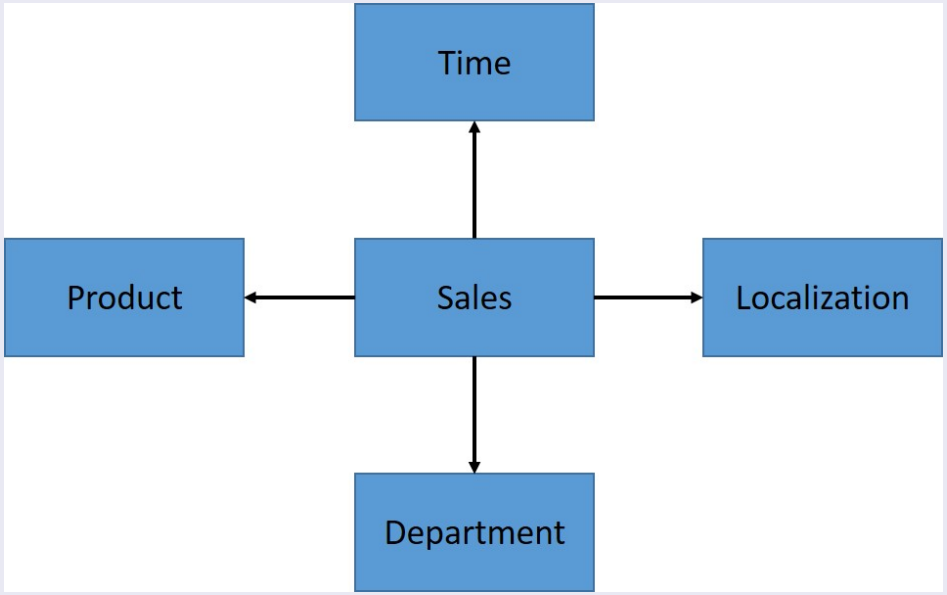
snr	cname	pname	qty	date
2	Smith	jeans	2	051012
3	Smith	shirt	4	101012
1	Ross	jacket	3	070812

$\texttt{JI}_{cname}$

cnr	snr
1	2
1	3
3	1

Abbildung: Verbund-Index für Relationen `Customer` und `Sales` über Attribut `cname`

Anwendungsbeispiel *Data-Warehouse*: Stern-Verbund zu einer *Fakten-Tabelle* `Sales` über 4 *Dimensionstabellen* mittels Bitmaps und Verbund-Index.



- Bilde jeweils einen Verbund-Index zwischen den Dimensions-Tabellen und der Fakten-Tabelle.

  Vermerke zu jedem Wert einer Dimensions-Tabelle die zugehörigen Fakten in Form eines Bitmaps.
- Werden nur Sales bzgl. gewisser Dimensionen angefragt, dann können diese Sales mittels boolscher Operatoren über den betreffenden Bitmaps bestimmt werden.

7.5 Optimierung

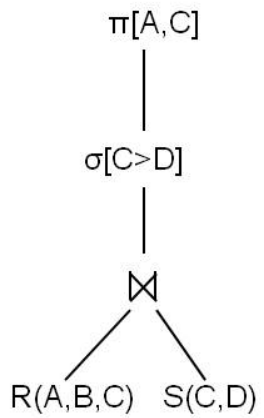
Optimierungstechniken

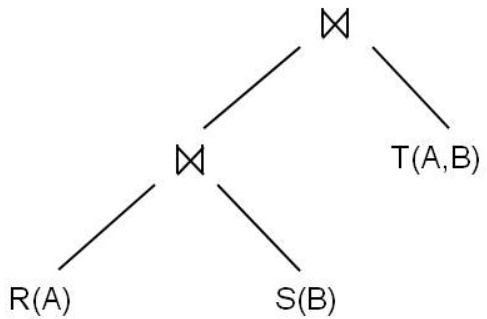
- **Semantische Optimierung**:
  Ausnutzen von Informationen über Integritätsbedingungen und (funktionalen) Abhängigkeiten.
- **Algebraische Optimierung**:
  Äquivalenzbewahrende Umformungen von Anfrageausdrücken.
- **Physische Optimierung**:
  Auswählen von Zugriffspfaden (Indexstrukturen).

7.5.1 Algebraische Optimierung

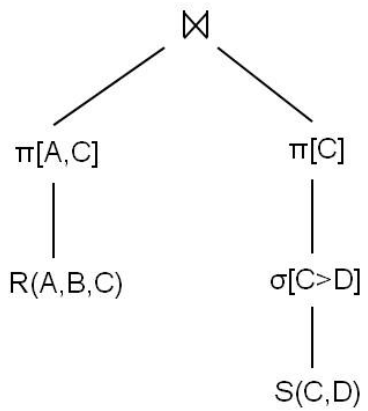
Anfragebäume zweier Algebraausdrücke:

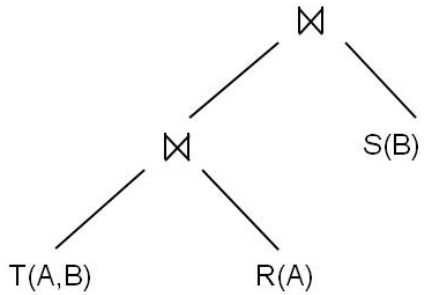
$\pi[A, C]\sigma[C > D](R(A, B, C) \bowtie S(C, D)), \qquad (R(A) \bowtie S(B)) \bowtie T(A, B))$





Äquivalenzerhaltende Umformungen obiger Ausdrücke:





Äquivalenz

Zwei Ausdrücke der Algebra Q, Q' heißen *äquivalent*, $Q \equiv Q'$, genau dann, wenn für jede Instanz $\mathcal{I}$ der Datenbank für die Antworten $Q(\mathcal{I})$ und $Q'(\mathcal{I})$ der Ausdrücke Q und Q' gilt: $Q(\mathcal{I}) = Q'(\mathcal{I})$.

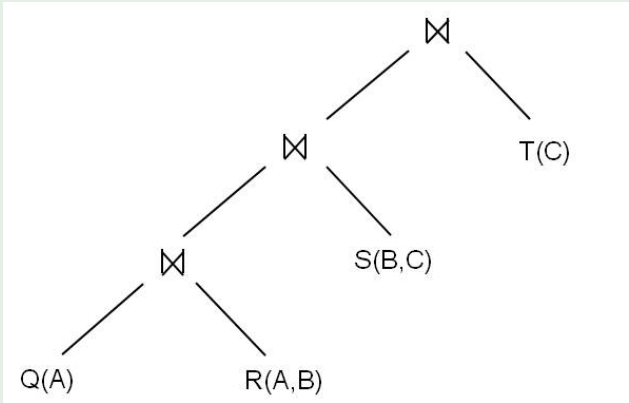
Beispiele

Sei attr(α) die in einer Selektionsbedingung α verwendete Menge von Attributen und seien R, S, T Relationsbezeichner mit Formaten X, Y, Z.

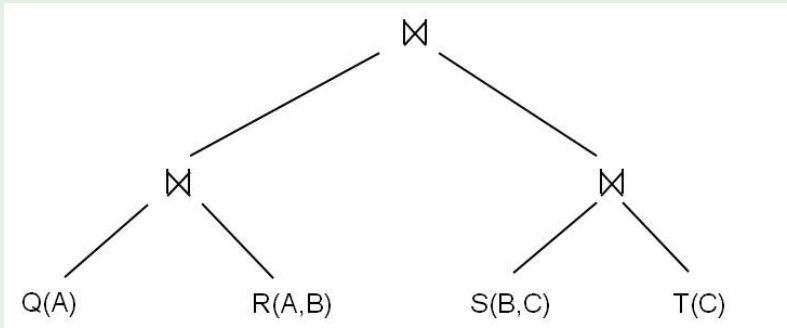
- $\bowtie$ ist assoziativ: $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
- $\bowtie$ ist kommutativ: $R \bowtie S \equiv S \bowtie R$
- $Z \subseteq Y \subseteq X \Rightarrow \pi[Z](\pi[Y]R) \equiv \pi[Z]R$
- $\text{attr}(\alpha) \subseteq Y \subseteq X \Rightarrow \pi[Y](\sigma[\alpha]R) \equiv \sigma[\alpha](\pi[Y]R)$
- $R \bowtie R \equiv R$
  Betrachte die Definition des Verbundes $R_1 \bowtie R_2$. Da hier $R = R_1 = R_2$ mit X Format von R, folgt $R \bowtie R = \{\mu \in \mathit{Tup}(X) | \mu[X] \in \mathcal{I}(R)\}$ für eine beliebige Instanz $\mathcal{I}(R)$. Damit gilt: $R \bowtie R = \{\mu \in \mathit{Tup}(X) | \mu \in \mathcal{I}(R)\} = \mathcal{I}(R)$.
- $X = Y \Rightarrow R \cap S \equiv R \bowtie S$
- $\text{attr}(\alpha) \subseteq X,\ \text{attr}(\alpha) \cap Y = \emptyset \Rightarrow \sigma[\alpha](R \bowtie S) \equiv (\sigma[\alpha]R) \bowtie S$

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

left-deep-tree Join-Auswertung



bushy-tree Join-Auswertung



7.5.2 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.
  - Selektivität der *Selektion*:
    $\sigma[\alpha](R)$: $sel_\alpha = \frac{|\sigma[\alpha](R)|}{|R|}$
    Spezialfall $\alpha : A = c$ und A Schlüssel von R?
    $\Rightarrow sel_\alpha = \frac{1}{|R|}$
  - Selektivität des *Verbunds*:
    $R \bowtie S$: $sel_{RS} = \frac{|R \bowtie S|}{|R \times S|} = \frac{|R \bowtie S|}{|R| \cdot |S|}$
    Spezialfall $R \bowtie_{R.A=S.B} S$ und A Schlüssel?
    $\Rightarrow sel_{RS} \leq \frac{1}{|R|}$ 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 $\bowtie T$).

  Dieses Prinzip nennt man *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.

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

1

[1] In: ACM SIGMOD Conference, 1979.