9 Auswertung von Anfrageoperatoren

9.1 Selektion

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

- Index zu A,
- Sortierung zu A,
- Operator op.

Auswertung von Formeln

- konjunktive Verknüpfung,
- disjunktive Verknüpfung,
- Normalform.

9.2 Projektion

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

- Duplikate entfernen?

9.3 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. Sei N die benötigte Anzahl Seiten für r und M die benötigte Anzahl Seiten für s.

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

Nested-Loop-Verbund

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

Varianten

- Block-Nested-Loop-Verbund.
- Index-Nested-Loop-Verbund.

  Index über Verbundattribute einer, bzw. beider Relationen,

  index-on-the-fly.

Anzahl Seitenzugriffe?

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.

Anzahl Seitenzugriffe?

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 h_1 eine Hashfunktion mittels der die kleinere der beiden Relationen r und s, o.B.d.A. r, so in Blöcke aufgeteilt werden kann, dass jeder Block in den Hauptspeicher passt.
- Wende h_1 auf r und s an.
- Bilde dann den Verbund unter Zuhilfenahme einer von h_1 verschiedenen Hauptspeicher-Hashfunktion h_2,
  - indem wir einen Block von r vollständig in den Hauptspeicher lesen,
  - dabei die Tupel in den Seiten des eingelesenen Blocks von r mittels h_2 auf einen Hauptspeicherbereich $H(r)$ streuen, und dann
  - die Seiten des zugehörenden Blocks zu s nacheinander in den Hauptspeicher holen und für jedes Tupel die korrespondierenden Tupel von r mittels h_2 in $H(r)$ bestimmen.
  - Iteriere das Verfahren bis alle Blöcke von r bearbeitet.

Anzahl Seitenzugriffe?

9.4 Mengenoperatoren und Aggregierung

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

Gleichheit der beteiligten Formate!

- $r \cap s = r \bowtie s$.
- $\cup$ und $-$?

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. Wann?

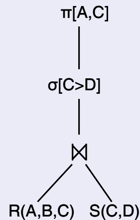
9.5 Optimierung

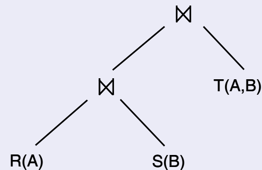
Optimierungstechniken

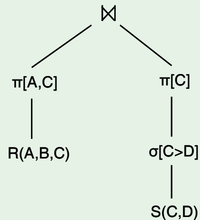
- semantische,
- algebraische,
- physische.

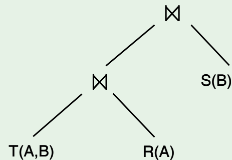
Algebraische Optimierung

äquivalenzerhaltende Umformung des Anfragebaums eines Algebraausdruckes









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?
  - $R \bowtie S$: $sel_{RS} := \frac{|R \bowtie S|}{|R \times S|} = \frac{|R \bowtie S|}{|R| \cdot |S|}$.
    Spezialfall $R \bowtie_{R.A=S.B} S$ und A ein Schlüsselattribut?

Pipelining

- Eine vollständige Materialisierung von Zwischenergebnissen wird vermieden.
- Zwischenresultate werden direkt an einen Folgeoperator weitergereicht.
- $\sigma[A = 5 \wedge B > 6]R$?
- $(R \bowtie S) \bowtie T$?

9.6 Auswertung von XPath

Enthaltensein-Anfragen

- Jeder Knoten eines XML-Baumes identifiziert einen Teilbaum und damit einen zusammenhängenden Ausschnitt des Dokumentes.
- Die zugehörigen Ausschnitte des Dokumentes sind ebenso in einander enthalten.
- XPath-Ausdrücke sind in diesem Sinn Enthaltensein-Anfragen, da mittels ihnen eine Menge von Knoten innerhalb eines XML-Baumes, bzw. eines Teilbaumes eines XML-Baumes, lokalisiert werden.
- Auswertung der `descendant`-, `ancestor`-, `following`- und `preceding`-Achse.

Haupt-/Nebenrang-Darstellung

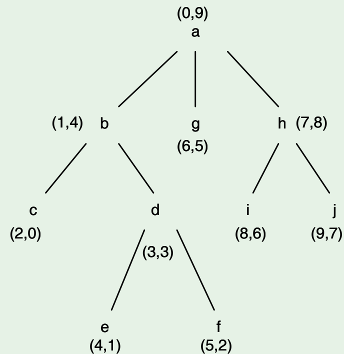
- Wir ordnen jedem Knoten v eines XML-Baumes T seinen *Hauptrang* (engl. *preorder rank*) $pre(v)$ und seinen *Nebenrang* (engl. *postorder rank*) $post(v)$ zu.
- Haupt-, bzw. Nebenrang der einzelnen Knoten ergeben sich aus ihrer Position bzgl. der Haupt-, bzw. Nebenreihenfolge innerhalb des XML-Baumes.
- Die Hauptreihenfolge ergibt sich, indem wir, beginnend mit der Wurzel, einen Knoten des Baumes *vor* seinen Kindknoten betrachten und die Kindknoten anschließend rekursiv in der Reihenfolge von links nach rechts.
- Die Nebenreihenfolge ergibt sich, indem wir, wiederum beginnend mit der Wurzel, einen Knoten betrachten, *nachdem* wir rekursiv seine Kindknoten von links nach rechts betrachtet haben.

Rangberechnung nach Haupt-/Nebenreihenfolge

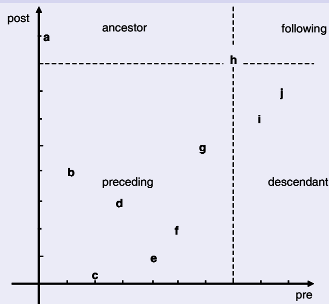
```
function Rang(node) {
    var children = node.childNodes;
    alert('Hauptrang zu '+node.id+': '+hauptRang);
    hauptRang++;
    for (var i = 0; i < children.length; i++)
       Rang(children[i]);
    alert('Nebenrang zu '+node.id+': '+nebenRang);
    nebenRang++;
}
```

Beispiel

```
<a>
  <b>
    <c>  </c>
    <d>
      <e>  </e>
      <f>  </f>
    </d>
  </b>
  <g>  </g>
  <h>
    <i>  </i>
    <j>  </j>
  </h>
</a>
```



Koordinaten-Darstellung



v' ist Element der `descendant`-Achse von $v \iff pre(v) < pre(v')$ und $post(v) > post(v')$,

v' ist Element der `ancestor`-Achse von $v \iff pre(v) > pre(v')$ und $post(v) < post(v')$,

v' ist Element der `preceding`-Achse von $v \iff pre(v) > pre(v')$ und $post(v) > post(v')$,

v' ist Element der `following`-Achse von $v \iff pre(v) < pre(v')$ und $post(v) < post(v')$.

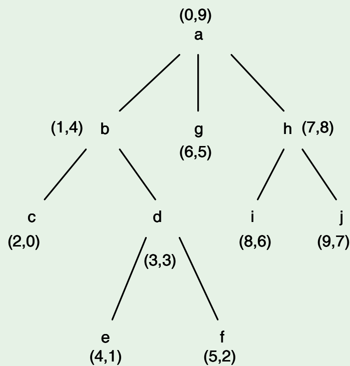
Relationale Darstellung

- Im Folgenden ausschließlich Elementknoten eines XML-Baumes.
- `Element(Haupt, Neben, Name)`; Haupt- und Nebenrang und Elementnamen.
- `Haupt` ist der Schlüssel.

Beispiel: d/descendant::*

Element

Haupt	Neben	Name
0	9	a
1	4	b
2	0	c
3	3	d
4	1	e
5	2	f
6	5	g
7	8	h
8	6	i
9	7	j




```
SELECT E2.Haupt
FROM Element E1, Element E2
WHERE E1.Name = 'd' AND
   E1.Haupt < E2.Haupt AND
```

Auswertung

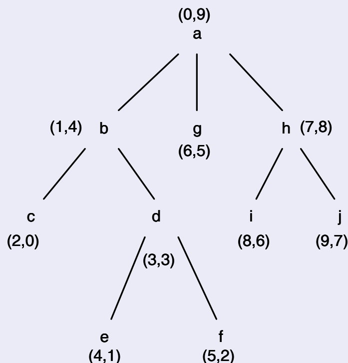
```
SELECT E2.Haupt
FROM Element E1, Element E2
WHERE E1.Name = 'd' AND
   E1.Haupt < E2.Haupt AND E1.Neben > E2.Neben
```

- Bestimme diejenigen Zeilen, die die Bedingung $pre(\mathtt{d}) < pre(v')$ erfüllen,
- Bestimme die Menge der Zeilen, die die Bedingung $post(\mathtt{d}) > post(v')$ erfüllen.
- Berechne den Schnitt beider Mengen.

Bei XML-Bäumen mit einer großen Anzahl Knoten können die als Zwischenergebnis berechneten Mengen erheblichen Umfang annehmen, so dass die Frage nach einer effizienteren Technik relevant wird.

Optimierung

Lediglich das *relative* Verhältnis der Rangwerte zueinander ist von Bedeutung und nicht die absoluten Werte.



Element^{opt}

Haupt'	Neben'	Name
0	19	a
1	10	b
2	3	c
4	9	d
5	6	e
7	8	f
11	12	g
13	18	h
14	15	i
16	17	j

Rangberechnung für Optimierung

```
function Rang(node) {
    var children = node.childNodes;
    alert('Hauptrang zu '+node.id+': '+RangZähler);
    RangZähler++;
    for (var i = 0; i < children.length; i++)
       Rang(children[i]);
    alert('Nebenrang zu '+node.id+': '+RangZähler);
    RangZähler++;
}
```

Descendant-Achse optimiert

v' ist Element der `descendant`-Achse von $v \Leftrightarrow$
$pre(v) < pre(v') < post(v)$, bzw. $pre(v) < post(v') < post(v)$.

H(d)	N(d)	Haupt'	Neben'	Name
4	9	0	19	a
4	9	1	10	b
4	9	2	3	c
4	9	4	9	d
4	9	5	6	e
4	9	7	8	f
4	9	11	12	g
4	9	13	18	h
4	9	14	15	i
4	9	16	17	j

Für die übrigen Achsen kann eine vergleichbare Optimierung nicht erreicht werden.

Beispiel /descendant::B/descendant::*

```
SELECT E3.Haupt
FROM Element E1, Element E2, Element E3
WHERE E1.Haupt = 0 AND E2.Name = 'B' AND
   E1.Haupt < E2.Haupt AND E2.Haupt < E1.Neben AND
   E2.Haupt < E3.Haupt AND
```

Beispiel /descendant::B/ancestor::A

```
SELECT E3.Haupt
FROM Element E1, Element E2, Element E3
WHERE E1.Haupt = 0 AND E2.Name = 'B' AND E3.Name = 'A' AND
   E1.Haupt < E2.Haupt AND E2.Haupt < E1.Neben AND
   E2.Haupt > E3.Haupt AND
```

Beispiel /descendant::A[descendant::B]

```
SELECT E2.Haupt
FROM Element E1, Element E2, Element E3
WHERE E1.Haupt = 0 AND E2.Name = 'A' AND E3.Name = 'B' AND
   E1.Haupt < E2.Haupt AND E2.Haupt < E1.Neben AND
   E2.Haupt < E3.Haupt AND
```